PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{5} : G01N 21/05, 35/00	A1	**(11) International Publication Number:** WO 90/05293 **(43) International Publication Date:** 17 May 1990 (17.05.90)

(21) International Application Number: PCT/US89/04981

(22) International Filing Date: 7 November 1989 (07.11.89)

(30) Priority data:
269,051 8 November 1988 (08.11.88) US

(71) Applicant: APPLIED BIOSYSTEMS, INC. [US/US]; 850 Lincoln Centre Drive, Foster City, CA 94404 (US).

(72) Inventor: MICHEL, Bruno ; Obstgartenweg 13, CH-8136 Gattikon (CH).

(74) Agent: SMITH, Joseph, H.; Applied Biosystems, Inc., 850 Lincoln Centre Drive, Foster City, CA 94404 (US).

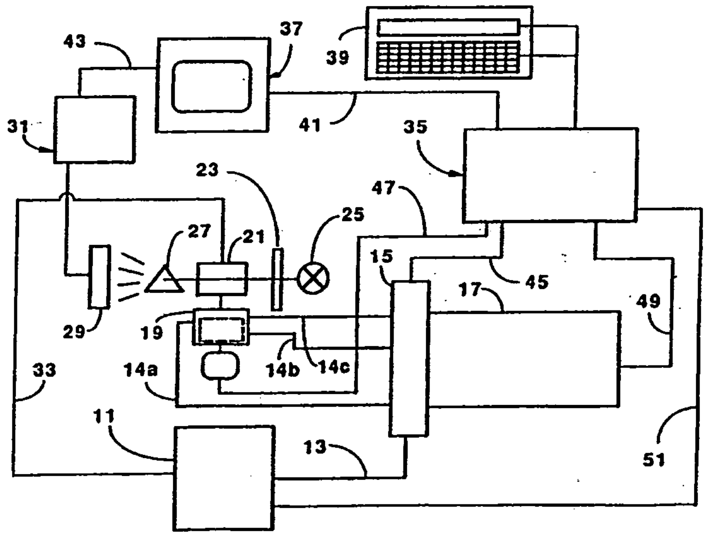
(81) Designated States: AT (European patent), AU, BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: ASSAYOMATE

(57) Abstract

An automated apparatus for monitoring chemical reaction assays which includes a supply system (15, 17) for providing accurately metered solutions of reactants to a mixing chamber (19). The mixing chamber (19) is connected to a reaction chamber (21) wherein a substantial portion of the reaction between the reactants occur. A physical parameter, which is a function of the concentration of at least one of the reactants and reaction products, is measured in reaction chamber (21). A computer (35 and 39) is used to automatically control the supply system, mixing chamber and reaction chamber and to analyze the data obtained from the reaction chamber (21) to determine kinetic constants and other parameters associated with the assay.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MG	Madagascar
AU	Australia	FI	Finland	ML	Mali
BB	Barbados	FR	France	MR	Mauritania
BE	Belgium	GA	Gabon	MW	Malawi
BF	Burkina Fasso	GB	United Kingdom	NL	Netherlands
BG	Bulgaria	HU	Hungary	NO	Norway
BJ	Benin	IT	Italy	RO	Romania
BR	Brazil	JP	Japan	SD	Sudan
CA	Canada	KP	Democratic People's Republic of Korea	SE	Sweden
CF	Central African Republic			SN	Senegal
CG	Congo	KR	Republic of Korea	SU	Soviet Union
CH	Switzerland	LI	Liechtenstein	TD	Chad
CM	Cameroon	LK	Sri Lanka	TG	Togo
DE	Germany, Federal Republic of	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		

ASSAYOMATE

Field of the Invention

The present invention is in the area of apparatus and methods for automatically combining chemicals to form reactions, and relates more specifically to mixing assays of enzymes, substrate solutions and buffer, and measuring characteristics of the ensuing chemical reactions. An important use is for the determination of kinetic constants for enzyme catalyzed reactions.

Background of the Invention

Biochemists in about the last quarter century have discovered the generalities and many of the details of the complex nature of the chemical reactions that take place in living organisms, as for example in the cells of human beings. Central to that chemistry is the synthesis of a large variety of proteins from the twenty known biological amino acids. In the complex process by which the genetic code on deoxyribonucleic acid (DNA) is copied and proteins are finally assembled according to the code, chemical reactions are catalyzed by other proteins called enzymes.

Because of the critical role of enzymes in biochemical processes, a large amount of attention and time is spent in chemical laboratories in the preparation and analysis of enzymes, and one of the critical issues is the measure of what are known as the kinetic constants of enzyme catalyzed reactions. Knowledge of the kinetic constants is important in quality control of enzyme preparations, screening of genetically engineered enzymes, drug research, kinetic research, and other areas.

An enzyme works as a catalyst by attaching to substrate molecules that are elements in the chemical reaction catalyzed, and bringing the elements into the close proximity required for reaction to occur. There are energy considerations in the process that take place to favor the direction of a particular reaction as well. The elements to which an enzyme attaches are called substrates of the enzyme, and every enzyme is specific to one or more substrates. The rate of an enzyme catalyzed process depends on the affinity of the enzyme for the substrate, the anount of the enzyme present, and the steady-state activity of a single enzyme molecule. With a very high substrate concentration the enzyme is saturated, and works at a maximal velocity vmax. At lower substrate concentrations the apparent velocity depends on the ability of the enzyme to bind its substrate(s) (affinity). The specificity is expressed by the ratio of an enzyme's affinity for one substrate as opposed to it's affinity for another.

The most relevant theory recognized in the art for the process of enzyme catalyzed reactions is the Michaelis -Menten theory, which assumes that an enzyme first combines with its substrate to form an enzyme-substrate complex, which then breaks down in a second step to form free enzyme and product. Based on this model the Michaelis - Menten equation was developed as the rate equation for reactions catalyzed by enzymes having a single substrate.

$$v = \frac{vmax * [S]}{km + [S]}$$

v is the reaction velocity, which is the change of substrate (or product) concentration per unit time, measured at concentration [S] of free substrate. Usually [S] is much larger than the enzyme concentration, and can be set equal to the total substrate concentration. vmax is the maximum velocity the substrate can achieve. km is the Michaelis constant, which is the substrate concentration at which the reaction rate reaches 50% of its maximal velocity.

Examination of the Michaelis-Menten equation shows that at low substrate concentrations the reaction velocity v is nearly proportional to substrate concentration, but as the substrate concentration is increased, the reaction becomes independent of the substrate concentration, and approaches a maximum assymptotically at vmax. The velocity never reaches vmax at finite substrate concentrations, so it is not accurate to estimate vmax by making a single measurement. km and vmax can only be determined by a relatively large number of measurements at varying substrate concentrations.

In addition to the kinetic constants for enzymes for the Michaelis-Menten equation, there are many other kinetic constants to be measured. An example is in the analysis of drugs, where a drug is expected to inhibit or activate one or more enzymes, so experimental determinations similar to the determinations relative to the velocity for an enzyme catalyzed reaction may be used to determine the inhibition constant ki or activation constant ka for a specific drug. For a more detailed presentation of the theory of enzyme catalyzed reactions and the Michaelis-Menten analysis, see one or more of many reference books on the subject of biochemistry, such as Biochemistry, Copyright 1975, 1981 by Lupert Stryer, published by W. H. Freeman and Company, San Fransisco, CA.

In a procedure to determine kinetic constants a relatively large number of assays have to be prepared and the reaction velocity measured as a function of changing concentration. The velocity of the reaction cannot be measured directly, but can be calculated or otherwise derived from measurements of concentration of an assay relative to time. This is usually done in what is known as a stopped flow apparatus by filling syringes with buffer, enzyme, and substrate (in the case of enzyme studies), simultaneously injecting the materials at predetermined rates or at a predetermined quantity into and through a mixing chamber, and into an observation container called a cuvette. The cuvette is fitted with spectophotometry equipment so that the optical density of the mixture may be monitored as a function of time. The optical density is known to change as a direct function of reaction.

Once an assay is mixed, and the chemical reaction begins, the concentration of substrate and product begins to change. The only correct velocity relative to substrate concentration, then, is the initial velocity. The procedure, therefore, is to monitor optical density of the reactants and product mixture for a short time, and plot the optical density (concentration) as a function of time. The initial velocity for the beginning concentration for this procedure is the slope of the curve at the beginning of the curve. Once a large number of velocity values at different concentrations have been derived, that data can be plotted, and the kinetic constants can be determined from the new plot.

In the process it is good practice to make measurements at each point more than once to statistically reduce the effect of errors. To accomplish such a procedure manually is a process that can take hours, because the data must be collected, correlated, plotted, and extended. Moreover, the chances for error in a manual procedure are quite high.

It is also sometimes necessary to measure the kinetic constants of an enzyme as a function of pH, ionic strength, inhibitors, activators, or a second substrate, increasing the quantity of measurements often to hundreds of assays.

Another problem with conventional techniques is that there is a definite time lag between the mixing of an assay and the measurement of optical density change. The lag can often be 5 seconds or more, and very fast reactions cannot be measured at all. It is also true that many enzymes are unstable, and errors are introduced through the degradation of stock solutions during the course of a determination procedure, especially if that procedure takes a long time to accomplish.

Another problem is that measurements with conventional apparatus are usually made at ambient conditions, and no effort is made to know the temperature, or to maintain a constant temperature during a procedure or series of procedures, even though reaction rates are known to be sensitive to temperature changes.

Still another problem accrues from conventional techniques by which measurements are made. For example, it is often necessary to dilute a stock solution of enzyme with a buffer solution to obtain a much lower concentration than the original. It is not unusual to dilute a solution by a factor of 1000:1 and more. Manual techniques with syringes are not adequate to making such measurements accurately.

Yet another problem with conventional techniques is that the solutions to be mixed may not always be easily miscible, and may vary widely in viscosity, so efforts to obtain homogeneity can result in time delays as well. Also, such solutions do not always mix intimately in spite of time delaying efforts, so measured results are often innacurate.

A certain amount of automation has been brought to the area of mixing assays and recording results. Robotic systems have been proposed and built for performing some of the manual operations. The time, accuracy, and mixing problems, however, have not been corrected. The robotic systems up to the present time have been severely limited in the range of concentrations that may be used, the number of assays that can be prepared and monitored without manual inervention, the accuracy of results, and the general flexibility of the systems.

What is needed is a completely automatic system for rapidly determining kinetic constants in enzyme catalyzed reactions, and for performing other similar procedures requiring combination of fluids under demanding conditions. Such a system needs to maintain the temperature of fluids delivered, accurately deliver from stock solutions, accurately mix the samples for measurement, begin measurements in a very short time after mixing, and provide homogeneity in mixing the stocks. in addition, such a system should provide a way to dilute original stock solutions by very large ratios, and do so accurately. Moreover, it has to be able to perform a wide spectrum of sample preprocessing, such as preincubating enzyme with inhibitor or a second substrate, and treating enzymes or substrates with various chemicals. The system needs also to be provided with suitable computer devices to facilitate the performance of the desirable features above, and to aggregate data and provide on-line analysis and display of the calculated and projected results.

Summary of the Invention

In accordance with preferred embodiments of the invention and apparatus is disclosed for automatically and rapidly determining kinetic constants in enzyme catalyzed reactions and which meets the needs described above for such an automated system. The apparatus includes a supply system for providing an accurately metered plurality of solutions containing chemical reactants which when combined have a defined kinetic constant associated with their reaction. A mixing chamber system is connected to the supply system for mixing the plurality of solutions from the supply system. In the preferred mode, a temperature control system controls temperature of the mixing chamber system and the fluid delivery system. A reaction chamber is provided where a substantial portion of the reaction between the reactants occurs, the reaction chamber being connected to the mixing chamber system. A detection system is used for measuring a physical parameter that is a function of concentration of at least one of the reactants and reaction products in the reaction chamber. A computer system coupled to the supply system, to the mixing chamber system, to the temperature control system, and to the detection system, is used for automatically controlling the supply system to provide a plurality of sets of simultaneous metered volumes of the solutions to the mixing chamber system, each set corresponding to a predefined ratio of the reactants, for automatically controlling the temperature of the mixing chamber system, and for automatically causing the detection system to make measurements at a plurality of times for each set of the simultaneous metered volumes to determine changes of the concentration in time for each set. The computer system is then used to analyze the data to determine kinetic constants and other parameters associated with the assay.

Particular features of the apparatus that are significant include a new and unique mixing chamber system that uses a floating magnetic head inside the mixing chamber which is magnetically driven from outside the chamber so that the inside of the chamber can be contaminant free. Further, the design is significant in providing an extremely small mixing volume which can still achieve very high mixing efficiency.

Another feature of the apparatus is use of an accurately driven syringe system, which in combination with an automated sample delivery system, and a versatile software control system, provides completely automatic operation for analyzing kinetic characteristics and enables enzymatic screening to be performed on a realistic and efficient basis.

Brief Description of the Drawings

Fig. 1 is a block diagram presenting important elements of the first preferred embodiment of the invention.

Fig. 2 is a perspective view of an arrangement of components of the first preferred embodiment.

Fig. 3 is a perspective view of an autosampler unit showing an automatic sample changer.

Fig. 4 is an elevation view, partially in section, showing a motor driven syringe unit.

Fig. 5 is a schematic of an arrangement of valves and syringe units in the first preferred embodiment

Fig. 6 is a broken section view showing the construction and operation of a mixer in the first referred embodiment.

Fig. 7 (A) is a section view of a mixer module including the mixer of Fig. 6.

Fig. 7 (B) is a section view of a connection between a fluid line and the mixer of Fig. 6 and Fig. 7 (A).

Fig. 8 is a face view of a display and keyboard module for operator control.

Fig. 9 illustrates a syringe control menu for control of syringe movements.

Fig. 10 (A) is a block diagram of software architecture.

Fig. 10 (B) is the Main Menu of the Kinetic Software.

Fig. 10 (C) is Screen 1 of a Measure Parameter Menu.

Fig. 10 (D) is Screen 2 of the Measure Parameter Menu.

Fig. 10 (E) is a setup menu for the Kinetic Software.

Fig. 10 (F) is a first screen of a Measure Menu.

Fig. 10 (G) is the second screen of the Measure Menu.

Fig. 10 (H) is a Setup Menu

Fig. 10 (I) is a Debug Menu.

Fig. 11 shows a printout of a typical setup for determination of Mikaelis constant for a particular exemplary enzyme using the apparatus of the invention.

Fig. 12 shows a printout of measurement data using the setup of Fig. 11.

Fig. 13 shows a plot of the derivative of the absorption as a function of time for different substrate concentrations for the exemplary enzyme.

Fig. 14 is a printout showing the different turnover numbers for different wavelengths of the detector system for the exemplary enzyme.

Fig. 15 is a plot of the turnover number versus substrate concentration for the exemplary enzyme.

Fig. 16 is a plot of turnover number for several determinations illustrating the reproducibility of the apparatus of the invention.

Description of the Preferred Embodiments

Fig. 1 is a block diagram showing equipment comprising an automated assay system, hereinafter called the "Assayomate." Shown there is a first preferred embodiment of the present invention, including an arrangement of fluid paths and control lines between the various pieces of equipment. An autosampler module 11 is a temperature-controlled reservoir region for presentation of stock and sample solutions such as buffer, substrate, and enzyme which the computer-automated system may use to develop assays of widely varying concentrations for measurement to determine kinetic constants. Stock solutions, like buffer solutions and water, that may be used in a wide variety of assay preparations, may be kept in the autosampler in relatively large quantities in reservoirs. Other solutions, enzyme solutions, inhibitors, and accelerators for example, are placed in probes in a rack that may be moved on a stage relative to a pipette that is robotically manipulated. In the first preferred embodiment the stage is a rotating, indexing carrousel.

Fluid transfer lines, represented as path 13, connect the reservoirs and the pipette in the autosampler to a valve module 15. The valve module is for switching sample and stock solutions to and from the autosampler and other parts of the apparatus.

An array of motor-driven syringes 17 is connected to the autosampler through valves in the valve module. The syringes are for drawing solutions from stock solutions and probes in the autosampler, and for feeding the drawn solutions in precise amounts at precise rates for assay preparation.

The driven syringes deliver solutions via valve module 15 to a unique motor-driven mixer 19 which is designed to insure a complete and rapid mixing of solutions, even with wide viscosity variations. Solutions delivered by means of the syringes go to the mixer through fluid-transfer lines between the valve block and the mixer, represented by lines 14a, 14b, and 14c. There is one line for each syringe, so there may be more than the three representative lines shown. The fluid transfer lines between the valve block and the mixer are routed side-by-side and within a larger tubing through which temperature-controlled water is circulated.

The mixture enters a cuvette 21 as a result of the forward drive of the syringes through the mixer, and action is stopped. After mixing and stop-flow, reaction between the solutions mixed has begun, and measurements may be made. In the case of velocity measurement for the determination of kinetic constants, a spectrophotometer is used to measure change in optical density relative to time, which is a measure of change in concentration of reactants and products from which reaction velocity values may be determined. In this case a shutter 23 opens and exposes a transparent opening through the cuvette. Light from a source 25 passes through the cuvette and the mixture in the cuvette. Then it passes through a prism 27 and impinges on a diode array 29. The changing signal from the diode array is delivered to a spectrophotometer control unit 31. Depending on the measurement to be made, other analytical instruments may be used, such as, for example, an electrochemical detector, a fluorescence photometer, or a chemiluminescence detector.

Different solutions are delivered to the mixer via dedicated lines from each syringe, and following measurement of an assay, waste is urged through a dedicated fluid line 33 back to a waste reservoir at the autosampler, making room for another assay in the cuvette.

A microprocessor-based control unit 35 is programmed to operate the assay apparatus, and a computer 37 provides additional, higher-level control functions. Although not shown in Fig. 1, the computer in the first preferred embodiment also has an associated printer, a keyboard for input, and a plotter attached. A dedicated keyboard and digital display 39 are a part of control unit 35. In an alternative preferred embodiment, microprocessor control unit 35 has a printer/plotter and a disk drive for storage. An RS-232 communication interface allows the computer to be connected to a larger host computer (not shown) for accomplishing tasks such as further data manipulation and display. In an alternative preferred embodiment, the microprocessor unit has a printer-plotter and disk drive for storage.

Computer 37 is connected to microprocessor module 35 by means of a communication bus through link 41 and module 35 is connected to the photometer control unit by means of link 43. In the first preferred embodiment the communication bus is an IEC bus. The microprocessor unit operates the syringe module through link 49, the valve module through link 45, the mixer apparatus through link 47, and the autosampler through link 51. Each of links 45, 47, 49, and 51 comprise more than a single electrical line, and some lines are for power while others are for control signals, such as microswitch closures, for example.

In the first preferred embodiment, autosampler 11, syringe module 17, valve module 15, and control unit 35 are mounted in a single cabinet, and the associated keyboard and display unit 39 are mounted in a face of the cabinet. The mixer and cuvette are a separate unit connected to the main cabinet by a tubing through which dedicated fluid lines are directed and through which temperature-controlled water is circulated to maintain sample fluids at a specific temperature prior to reaction. The water control unit is not shown in Fig.

1. By virtue of the separation, the mixer module can be inserted into various commercially available spectrophotometer devices and other analytical instruments.

The apparatus of the invention is used to study biochemical reactions, such as the determination of kinetic constants in enzyme catalyzed reactions, which cannot be measured directly for the reasons described above in the Background of the Invention. The method has to employ typically a large number of measurements of a dependent variable while varying an independent variable.

The syringes are sized so that a single ingestion of the necessary solutions from the reservoirs and probes in the autosampler is enough to mix and inject all of the assays necessary to determine the kinetic constants for a particular enzyme and substrate. This is important for two reasons: First, a refill during the determination of a kinetic constant may introduce a systematic error due to enzyme degradation. Second, if no refill is needed, the autosampler unit is free to perform other functions while the determination is still in process.

The keyboard unit associated with control module 35 can be used to input specific commands in a manual mode and to initiate programmed subroutines, as well as to initiate automatic determinations. Among many programmed procedures, there are procedures for wash, for loading stock materials, and for injection of the materials to form assays of varying concentrations to generate data for determination of kinetic constants for a specific enzyme.

The amounts of enzyme, substrate and other solutions for measurements of a data point are determined by the software and appropriate signals are generated through the microprocessor unit to cause the specified mix to take place by selected drive of the several syringes. The materials are urged to and through the mixer, and the mixer is driven to ensure an homogeneous mix of the materials. The mixture enters the cuvette very rapidly, and measurements are made with the analytical instrument dedicated to the particular task. In the case of a spectrophotometer measurement for velocity data, the shutter is opened to allow the photometric sequence to take place, during which data from the diode array is sent to the photometer conrol unit. Each data point may be repeated to establish accurate measurement, and sequencing continues for new data points until the needed amount of data is acquired to accurately determine the desired kinetic constants for an enzyme.

The speed of the mix and full preparation for a single data point is within 0.3 second, as opposed to typically 5 seconds or more for manual methods. Hence, fast reactions can be characterized. Also, sequencing is rapidly accomplished, so determinations that generally take hours manually can be performed in minutes with the automatic apparatus of the invention.

Fig. 2 provides an overview of the apparatus of the first preferred embodiment in order to better explain the organization of the units comprising the system shown in Fig.1, and to illustrate other elements. Autosampler 11 is a part of main cabinet 53, and microprocessor unit 35 with keyboard and display unit 39 is mounted in the same cabinet. The syringe array and valve block are also mounted in cabinet 53, although not seen in the view of Fig. 2.

A mixer module 20 which includes mixer 19 is separate and connected to cabinet 53 by tubing 55 through which the various fluid lines pass. Control and power lines are arranged alongside tubing 55, and temperature-controlled water passes through the tubing as well. The length of connecting tubing and power and control lines to the mixer module are such that the module may be mounted in an analytical instrument, such as a commercially available spectrophotometer devices, such as a Hewlett-Packard Model 8451 diode-array spectrophotometer, which is used for data collection in the first preferred embodiment for determination of kinetic constants.

Constant temperature water is supplied and maintained by a commercial unit 57. Computer 37 in the first preferred embodiment is an HP 85 computer and has a printer 61 and a plotter 63 attached. The HP 85 computer in some instances may be connected to a more powerful host computer, such as an HP 150, for more extensive data analysis, and there is an RS-232 interface for the connection. There are, in addition, Peltier units (not shown) for controlling temperature in the apparatus, and electronic units to control the Peltier units. The entire connected system is small enough to be arranged on a desktop.

In an alternative preferred embodiment, the main cabinet is more extensive, and encompasses a built-in parallel processing computer, the water temperature control unit, the spectrophotometer or other analytical instrument, and an internal printer-plotter and disk drive, as well as electronic controls for Peltier elements. This instrument may be remotely controlled via an RS-232 or IEEE-488 bus using software which runs on a Mac II or an IBM PS/2.

Autosampler Module

Fig. 3 shows a part of Autosampler Module 11 within cabinet 53 of Fig. 2. The Autosampler Module may be accessed through a door from outside the cabinet, and the enclosure is a temperature-controlled region by means of a Peltier unit (not shown) mounted on a side of the enclosure. A typical temperature inside the Autosampler enclosure is 4 degrees Centigrade.

Two stock reservoirs 65 and 67 rest on a shelf 69 within the cabinet. Reservoir 65 has a tubing 71 that passes from the reservoir thru a back wall of the unit, and reservoir 67 has a similar tubing 73. The tubings are flexible, small-bore fluid transfer tubes, and are connected at the other end to ports at the valve module. The stock reservoirs are for storage of stock solutions, such as substrate, inhibitor, or other chemical reagents. A solution can be changed or replaced if needed by lifting the tube from a reservoir and switching containers. As a general practice, the stock reservoirs are used for solutions that may be common to a large variety of experiments. Although only two such reservoir containers are shown in Fig. 3, a larger number may be placed in the Autosampler, each with a dedicated fluid tube. In the first preferred embodiment there are typically two to four stock reservoirs, for use as needed.

In addition to the stock solution reservoirs, there is an automated sample changer 75 for presenting different sample solutions automatically. An indexing stage 77 extending through shelf 69 is driven from below by a stepper motor 79 so that each of several probes, such as probe 81, supported by the stage, may be presented to a pipette 83. The pipette is carried by a slide carriage 85 in a guide frame 87, and the slide carriage is driven up and down by another motor drive below (not shown). The pipette is raised to clear a probe to allow the indexing stage to turn without interference, and lowered when a probe is in position.

Pipette 83 is connected to a fluid transfer tube 89, and the other end of the tube connects to an opening at the valve module so that probes may be drawn into one or another syringe of the syringe module through the valve module. An upper limit switch 91 signals the uppermost position of the pipette translation, and a lower limit switch 93 signals the lowermost position of the pipette translation. There are other limit switches (not shown) for signalling the presence of a probe rack on the indexing stage, and for monitoring the index position of the rotary stage.

Only four probes are shown supported by stage 77 in Fig. 3, for purposes of illustration, but there are many more in the Assayomate. Typically, there are as many as 50 probes arranged in circular rows. The sample changer allows a wide variety of sample solutions to be prepared in advance and presented to the automatic assay system, providing for maximum flexibility, effeciency, and productivity in determining kinetic constants, and also allows for empty containers to be provided to recieve processed samples and other effluent. The presence of empty containers also allows for mixing of sample solutions to different concentrations before introduction to the mixer in an analytical procedure. For example, if the initial concentration of a enzyme is too large, the enzyme can be diluted accurately using an empty container and using the syringes to control the dilution. An additional Peltier unit is incorporated into a part of the rotary stage, typically covering about 25% of the stage area for supporting containers, and is used to maintain certain sample solutions at a temperature other than the ambient selected temperature within the Autosampler enclosure. There is typically a wash station incorporated on the rotary stage, which is a position supporting a relatively large container for water or other reagent to be used to wash syringes and flush lines when desired.

Syringe and Syringe Drive Unit

Fig. 4 is a sectioned view of a single syringe unit 95 that is one of several units that comprise syringe array 17 in the first preferred embodiment. A Hamilton syringe 97 is operated by a motorized drive to draw solutions from probes and stock reservoirs in the autosampler module, through valves in valve block 15, and to inject solutions as required to mixer module 19 through the valve block after switching the fluid routes by switching the valve positions. Fluids are drawn and expelled through fluid tubing 99 connected to the syringe. A water jacket 101 surrounds the syringe, and temperature controlled water is passed through the jacket via tubes 103 and 105 which lead to suitable manifolding (not shown) which is in turn connected to water supply and control unit 57 (Fig. 3). There are three different sizes of syringes used in the first preferred embodiment of the Assayomate, one holding a charge of 50 ml, one of 5 ml, and one of 2.5 ml. The considerable difference in size is provided to allow a considerable difference in volume of different sorts of solutions needed to comprise an assay, and the volume difference is accomplished by varying diameter rather than stroke, so drive units can be common.

The plunger shaft 107 passes through a frame plate 109 and is attached to a drive carriage 111 by means of a conventional set screw 113 in the first preferred embodiment. Carriage 111 has an adjustable bearing slide 115 attached which guides on a guide rod 117. The guide rod is fixed in upper plate 109 and in a lower frame plate 119. Carriage 111 has a nut 121 attached at one end, and a threaded shaft 123 engages the nut. Motor 131 drives shaft 123 through a reducer 129, resulting in a very small rotational movement of the threaded shaft for a larger rotation of the motor, and hence a very small linear movement of the syringe plunger for motor rotation. In the first preferred embodiment a stepper motor is used so a specific volume of solution may be related to a single motor pulse, depending on syringe size. Plates 109 and 119 are spaced apart by a side plate 133, forming a rigid frame unit for the syringe unit. A microswitch 135 is fastened to plate 133 in a position to sense and report the forwardmost travel of the syringe plunger.

In the Assayomate, all of the syringes have a stroke of about 6 cm. The lead of the threaded shaft and nut is 1mm. per turn, and the gearing ratio is 16.66 : 1. The stepper motor requires 48 pulses for a single revolution, so a full stroke of a syringe requires 48,000 stepper pulses. With syringes of 50 ml, 5 ml, and 2.5 ml, the volume delivered (or drawn in) by a single pulse is 1 micro-liter, 0.1 micro-liter, and 0.05 micro-liter respectively. Typically, for determination of a kinetic constant, a substrate syringe has to perform minimally a stroke of .005 mm. and maximally a stroke of 10 mm. within 1 second.

In the first preferred embodiment there are typically three or four syringe units comprising syringe array 17, and the framing is designed so that the discrete syringe units may be mounted side by side in a rack-mount fashion in cabinet 53. Tubing from each syringe connects to specific ports of the valve block, water lines connect to suitable manifolds, and there are power and control wiring connections from each syringe unit to the microprocessor control unit. In other embodiments there may be a different number of syringes in the array.

<u>Valve Block</u>

Fig. 5 is a schematic showing the valve block in the first preferred embodiment, showing also the connection to the syringe array and the autosampler module. For this example, four syringe units are depicted. There are 8 solenoid activated valves in the valve module, two serving each of the four syringes, and all are shown in Fig. 5 in the deactivated state. The valves used are LFYA 120 and LFYX valves from the Lee Company of Westbrook, CT., although there are other suitable valves commercially available. The Lee valves have a very small internal volume (7.4 ml for the LFYA) and inert internal surfaces, both of which are advantages. Operation is at 12V and switching is accomplished within 10 msec.

Valve 151, in the deactivated state, directs fluid from syringe 153 to mixer 19; valve 147 deactivated directs fluid from syringe 149 to the mixer; valve 143 deactivated directs fluid from syringe 145; and valve

137 deactivated directs fluid from syringe 139 to the mixer. To fill or wash any one of the syringes, the pertinent valve 151, 147, 143, or 137 has to be activated.

When valve 151 is activated and valve 167 remains deactivated, syringe 153 is connected to a reservoir in the autosampler module through line 169; when valve 147 is activated and valve 163 remains deactivated, syringe 149 is connected to a reservoir in the autosampler unit through line 165; when valve 143 is activated and valve 159 remains deactivated, syringe 145 is connected to a reservoir in the autosampler through line 161; and when valve 137 is activated while valve 155 remains deactivated syringe 139 is connected to a reservoir in the autosampler through line 157.

Fluid line 171 is connected to pipette 83 (Fig. 3) at the automatic sample changer in the autosampler module, and splits to four lines, one each going to valves 155, 159, 163, and 167. By this arrangement, any one of the syringes may draw fluid from a probe on the stage of the automatic sample changer, or return material to a container on the stage. In the first preferred embodiment, syringe 145 typically draws from the automatic sample changer pipette through valves 143 and 159 activated, although this is but one of the many arrangements that can be made by switching valves off and on. In Fig. 5, lines from each of valves 137, 143, 147, and 151 extend to mixer 19. In operation some or all of the syringes are filled with fluid samples that together make up an assay. For example, syringe 139 may draw in buffer, syringe 145 substrate solution, and syringe 147 an enzyme solution from the automatic sample changer. By driving the syringe pistons forward simultaneously at programmed rates and distances, a pre-programmed amount of each of the components of an active assay are delivered to the mixer, where the enzyme catalyzed reaction begins immediately. The mixer assures complete homogeneity of the mix, which passes through the mixer into cuvette 173, where data is collected by means of a spectrophotometer or other analytical instrument. The syringes are of a size that a single draw is adequate in most cases to provide all of the assays needed, in some cases 100 or more, to determine accurately the kinetic constants for a particular enzyme and substrate. Effluent (completed assays) is forced on through the cuvette, and eventually back to a waste container at the autosampler unit.

The availability of variable probes and as many as four stock solutions in dedicated reservoirs allows for a very broad range of assays to be mixed and measured, and constants can be automatically determined for several enzyme and substrate combinations without pause for major service. The larger syringe 139, allows for such operations as flushing and cleaning of the lines and equipment.

The fluid line connections shown in Fig. 5 are exemplary of those of the first preferred embodiment, as are the number of valves and syringes. Such arrangement of the Assayomate is particularly advantageous for procedures for determining the kinetic constants of enzyme catalyzed reactions. There are many other advantageous ways that the connections might be made to accomplish other purposes in other embodiments, and the number of syringes and valves may be different as well.

Mixer and Measurement Module

Fig. 6 is a broken section illustrating the construction and operation of mixer 19 shown elsewhere in Figs. 1 and 5. This drawing is meant to show the general arrangement of detail elements to one another rather than specific construction of each element. A generally cylindrical outer body 175 is a fluid -tight enclosure with four incoming ports, and these each are connected to one of the lines 141, 177, 179, and 181 from valves 137, 143, 147, and 151 respectively. There is one outgoing port 183, which passes mixed fluids to the cuvette for measurement. The incoming lines penetrate body 175 around the periphery, so that incoming fluid from each of the lines (injected by the syringes) will enter the mixer in an annular space 185 between the outer body and an internal cylindrical drum 187. The tubes need not be perfectly orthogonal to the mixer body or exactly equally spaced as shown in Fig. 6, but it is important that they enter the annular space, and it is advantageous that they enter near the end of the mixer away from outlet line 183. In the first preferred embodiment, the body is inert plastic material, such as fluorocarbon material, and the drum is the same plastic material with imbedded ferromagnetic elements, although there are a number of other suitable materials that may be used. It is advantageous that the material be inert with the reagents used with the apparatus.

Drum 187 floats within the outer body of the mixer and is driven rotationally within the outer body by an external magnetic drive (not shown in Fig. 6) that also urges the drum against the top of the enclosure. By virtue of being always filled with fluid during use, there is always a film of fluid between the drum and the body, serving as a bearing surface, preventing galling and wear of the body and drum. In the first preferred embodiment, there are machined blades, such as blade 189, on the end of the mixer drum away from the end where the tubings enter, and arcuate grooves 191 in the end of the body opposite the blades on the drum. The drum is shorter than the cylindrical cavity of the body, leaving a space 193 between the drum and the end with the grooves.

In the processes of mixing assays and making measurements to determine kinetic constants, the materials used, such as enzyme solutions, are often quite expensive. The equipment is made small to require only small quantities to be used. The lines to the mixer from the syringe are less than one millimeter in diameter, and the diameter of line 183 is such to produce an exit path approximately equal in cross-sectional area to the total of the incoming lines, so there is no restriction at the outlet.

In operation, drum 187 is typically rotated about its central axis at a rotational velocity of from about 1000 to 10,000 revolutions per minute while small amounts of fluids are injected into annular space 185, less than .2 mm in width. The rotational speed is adjustable (programmable) through the software associated with the computer control system. The translation speed of the drum surface rotationally should be typically about 20 times the translational axial velocity of liquid mixture in the annular space to assure adequate mixing of solutions under all expected conditions. In some cases, one fluid may have a viscosity very much larger than another, and in many cases the required amounts of fluids may vary by as much as 2000 to 1. The rapid shearing action in the anullar space during injection assures that homogeneity is accomplished extremely rapidly, while only minimally slowing the flux of material through the mixer. Thus, the liquids can be injected with low pressure, unlike known passive mixing devices. In most cases, the injection step and complete mixing is done within 1 second. The dead time of the apparatus, which is the time between mixing and measuring, varies from .075 seconds to 1 second, due to different injection volumes and injection speeds. For assays with low substrate concentration the deadtime is kept as short as possible because substrate is used up quickly by the enzyme reaction, whereas for assays with high substrate concentration the dead time can be longer.

In the first preferred embodiment annular space 185 has an axial length of about 8mm and the gap width is about .2 mm. The total active volume of this annular space is thus about 25 microliters. During normal operation, 400 microliters are injected in 1 second. Under these conditions, materials remain in the shear field for about .0625 seconds. The axial velocity is 128 mm per second, and for the peripheral speed of the drum to be 20 times the axial fluid velocity, the rotational velocity of the drum must be more than 6000 revolutions per minute.

As fluids are injected and pass through the mixer, additional turbulent mixing is accomplished in space 193 by virtue of the blades, and the rotational direction of the mixture relative to the grooves helps to guide the material to the end orifice and into exit line 183. Space 193 is especially useful if the stepper motor driven syringes add volumes in a noncontinuous manner. In this case the space acts as a buffer zone to level out sharp pulses to a more steady concenteration.

Fig. 7 (A) is a section view of mixer module 20 (also shown in Fig. 2) which includes mixer 19. Mixer 19 is driven in the module by magnetic drive rotor 195 which is driven by motor 197. The motor, the magnetic rotor, and the mixer are located in separate compartments of an inner structure 199 of the module, and the magnetic nature of the mixer drive does not require a penetration of the mixer volume. The only physical penetrations of the mixer are the fluid line penetrations.

The mixer module has an outer shell 201, and the space between the inner and the outer shell is a water jacket for temperature-controlled water, circulated to control the temperature of solutions delivered in the fluid lines. Connector 205 is for connection to tubing 55 (Fig. 2) within which other fluid lines are routed. Line 141 is one of the lines from a valve in the valve block to the mixer, and is shown extending behind the inner shell and connecting to the mixer at connection 207. In the first preferred embodiment there are four lines from the valve block to the mixer for injection of solutions to form assays for measurement, but only one is shown in Fig. 7 (A) to avoid confusing clutter on the drawing.

Fluids injected are mixed in mixer 19 and the mixture is forced through the exit line and connection 209 into cuvette 21, where flow is stopped by virtue of the further injection of fluids being stopped, and measurement of optical density vs. time is made using the spectrophotometer (not shown in Fig. 7). Typically readings from 10 photodiodes are taken at 100 msec intervals, and the time window is adjustable through an input variable. The total length of time of measurement needs to be different for different substrate concentrations. Once measurement is complete, further syringe action is initiated, and the mixture in the cuvette is displaced through connector 211 and into line 213.

In the first preferred embodiment, a second cuvette 215 is mounted within the water jacket to the inside sidewall of the outer shell, and an analytical instrument 217 may be mounted in a well from outside the mixer module into this cuvette. Instrument 217 is typically either a temperature measuring instrument or an instrument for measuring acidity (pH). Effluent urged through the second cuvette exits the mixer module via fluid line 33 back to the autosampler module, where waste is captured in a reservoir for that purpose.

In addition to the fluid tubes described passing through tubing 55, there is additionally a supply tube for incoming water at a controlled temperature, which empties into the water jacket. The circulating water flows around the water jacket surrounding the mixer and the fluid feed lines, and backflows through tubing 55, returning the water to a manifold (not shown) that routes it back to water control unit 57. The overall height of the mixer module is about 8 cm., and the diameter is also about 8 cm. The various parts are made to fit together with suitable o-ring seals and fasteners, as is known in the art, to be demountable for service. Control lines and electrical power lines are typically routed to the mixer module outside and alongside tubing 55, and the lines are such that the module extends about 25 cm. from the valve module on a flexible tether, and may be easily moved and repositioned within the limitations of the line length.

Fig. 7 (B) is an enlarged section view of connector 207 where fluid line 141 from valve 137 joins the mixer body. It has been found that during stopped flow, some of the fluid in an incoming line to the mixer can be washed out into the annular space, which may reduce the accuracy of the next injection for the next assay. A unique connector design reduces this leakage to a minimum. An o-ring 235 is trapped against a shoulder of body 199 and pressed against the shoulder by means of a threaded cylinder 233 that surrounds fluid line 141, such that by adjusting the amount of engagement, the diameter of opening 237 is changed. Opening 237 will be a minimum during stopped flow, but under the influence of injection pressure while fluid is injected, will expand. The volume that can be washed out during stopped flow is reduced to the volume 239 of line 141 between the annulus of the mixer and the o-ring restricted position. In the first preferred embodiment the length of this volume is 1mm and the inside diameter of the tubing is .3 mm, so the wash-out volume is minimized to less than .1 microliter. The adjustable o-ring connection is typical of all of the tubing connections to the mixer, not just the connector shown for illustration.

There are several important features of the mixing system that bear emphasis. First, a major difference between this mixer system and the prior art is that the Assayomate system lacks bearings and seal on the rotating mixing body. For that rason the mixer can be built with absolutely inert surfaces and is essentially sevice free. There is no contamination with lubricant or other substances in the mixer. Second, the driving magnet fulfills three functions. It drives the mixing body, it pulls it ot one end of the mixing chamber, and it constantly adjusts the position of the mixing body, keeping it in the center of the mixing chamber. The design further allows a very narrow shear slit, which although is 0.2mm in the illustrated preferred embodiment, can probably be reduced to 0.05 mm. This is possible because the driving magnet and the fluids tend to center the mixing body, the mixture being a lubricant because of the Teflon outer coating of the mixing body. Also, the secondary mixing chamber below the shear mixing area allows pulsed operation of the stepper motor drive systems without causing pulses in the output of the mixing chamber. Also, the design of the input tubing with elastic orifices is unique in preventing leakage from the supply lines during stopped flow operations without interrupting flow during other times. Finally, the system allows the active volumes to be very small compared to typical prior art mixing systems.

Microprocessor Control Unit

Microprocessor unit 35 in the first preferred embodiment comprises a Z-80 microprocessor in a single-task operating system. It treats driving the syringes as a single-task procedure. It also comprises Read-Only memory (ROM), Random-Access memory (RAM), and a variety of other elements for accomplishing direct control of the machine elements of the apparatus. The other elements include such devices as tranceivers, decoders, stepper motor drivers, logic elements, and the like. All of the electronic elements are mounted on printed circuit boards mounted in cabinet 53.

In building and testing the apparatus of the first preferred embodiment, it was discovered that accuracy of results is better achieved when timing inaccuracy does not exceed 10 microseconds between syringe drives, and that the accuracy is limited with a single microprocessor. For this reason, in an alternative preferred embodiment, a microprocessor is dedicated to each of the syringe drives and the dedicated microprocessors are slaved to an additional microprocessor, allowing parallel control functions to be performed during the driving of the syringe units.

Operation of the apparatus includes indexing of the rotary stage of the automatic sample changer, lifting and lowering of the pipette at the sample changer, operation of the solenoids of each of the valves in the valve block to switch the direction of fluid flow, independent drive of each of the stepper-motor driven syringe units of the syringe module, operation of the motor-driven mixer drum of the mixer module, and operation of the spectrophotometer unit. All of these control functions are performed through the microprocessor control unit, and the status of various parts of the apparatus is monitored through the position of microswitches. In the first preferred embodiment, the microprocessor used is a Z-80 microprocessor. There are, in addition, a number of data transfer, manipulation, and output functions performed by computer elements.

Keyboard and Display

Keyboard and display module 39 in the first preferred embodiment is built into the front of cabinet 53 as seen in Fig. 2. Fig. 8 is a closer view of the module, which consists of an LED status display 221 that is, in the first preferred embodiment, designed to show the positions of the valves and limit switches in real time; a dot-matrix LCD digital display 223, and a matrix of 32 keys in two 16-key patterns. The keys are for inputs to the control unit, and the two-line digital display is for messages and for display of each of a number of control menus that are used for input of variables at appropriate times, and for directing the flow of control programming. Many keys are dually identified, once on the key and once above. For example, the "B" key is also used for "J", and there is a shift key to accomplish the differentiation. The output shown on the digital display in Fig. 8 is the first two lines of a Syringe Control Menu.

Operation and Control

There are two levels of control in the overall control system: an automatic mode in which the syringes are moved together to inject fluids from the syringes in harmony, and in predetermined amounts, to the mixer for producing assays for measurement, and in which new probes may be introduced and all the serial operations necessary may be performed to complete entire series of assays for the determination of kinetic constants and for other analytical procedures; and an unsynchronized mode in which the syringes may be moved one after another, and other operations may be performed individually at the initiation of an operator for various reasons, such as cleanup, set-up, simple photometric measurements, metering amounts into sampler tubes, maintainance, and the like.

The unsynchronized mode is accomplished in the first preferred embodiment through system software programmed in Z-80 assembly language and resident at the microprocessor module in ROM. In this mode a user can enter a comprehensive set of single keystroke commands from the built-in keyboard, controlling all of the functions of the apparatus in a manual mode. For example, the "F" keystroke activates a function called "EMPTY". This function empties the currently active syringe back to the reservoir immediately. The message: "EMPTYING SYRINGE" is displayed on the built-in dot-matrix display during the time that the syringe is active. The complete set of keystroke commands is presented in Appendix A. The manual-mode operations in the first preferred embodiment are divided into three groups for control of the functions of the detector instrument, the automatic sample changer, and the syringes of the syringe array. The groups are accessed through prefix keys: (-) for the detector, (+) for the sample changer, and (*) for the syringes. With the prefix, softkey labels appear on the display, arranged to correspond to the eight keys labeled "A" through "H" on the front panel. Pressing the corresponding key initiates the function. Additional functions in each group are accessed through the shift key.

An important feature is that the primitive functions shown in Appendix A, which encompass all of the physical operations of the apparatus, can be performed either by keystroke command under operator initiation, as described above, or by command from an external CPU, such as the HP-85 computer, over a standardized digital interface, such as RS-232 or IEEE-488. The latter mode, with commands from the computer, is the automatic mode. The software for programmed operations is written in the first preferred embodiment in HP BASIC, and is resident at the HP 85. Operation can be switched at any time between the keypad (local - keystroke "U") and programmed operation from the HP-85 (remote - keystroke "R"). This allows maximum flexibility for troubleshooting and other manual procedures. In an alternative preferred embodiment, in which the master processor is an 80286, the automatic operations are controlled by programming written in "C" language, which is resident at the master processor. In this alternative preferred embodiment there is also a built-in computer in the main cabinet, as described above, and higher-level automatic functions are programmed in "C" on the built-in computer.

Determination of kinetic constants according to the model of the Michaelis-Menten model as described in the background of the invention is one of a number of useful applications of the present invention, and the Michaelis-Menten model of kinetic behavior is one of the models for data manipulation and display. The kinetic software for collecting reaction velocity data is useful for a broad range of kinetic models, some of which are programmed and selectable by the user at his option. In addition, there is an equation interpreter included in the software so that a user can enter other models of his choosing.

Syringe constants and updated status are stored in predetermined control memory blocks, and another memory block is designated as the "active" block. The constants and status include position of the piston in pulses; maximum pulses for a full length move; number of wait cycles after a pulse; number of functions executed for functions B, C, and D; maximal fill level in pulses; and the conversion factor for the particular syringe for motor pulses into volume. Whenever the control is not busy with a higher priority function, a Syringe Control Menu may be displayed. This menu is also displayed while in the local mode, so syringe data may be followed by an operator.

The complete Syringe Control Menu for the first preferred embodiment is shown as Fig. 9. Two lines at a time may be displayed, and other lines are displayed by rolling the menu up or down with the "up arrow" and the "down arrow" keys. There are in addition to the information lines on the window, "softkey" lables presented by pressing the space key. Pressing the softkeys in the local mode will activate the named function (see appendix A). The softkeys in the first preferred embodiment are the keys on the front panel labeled "A" through "H", and correspond respectively to the softkey designations f1 through f8.

The Kinetic Software

The User Software for determining kinetic constants is called the Kinetic Software in this specification. It is a menu-driven program set, and the various menus are shown by level in Fig. 10 (A). For more detailed descriptions, the Software documentation and software listings are appended.

The main program of the Kinetic software is started when the command "/" is received by the operating system. The first time the main program is called, all system parameters and variables are loaded with a set of user-definable default values. After the completion of these tasks an initialization flag is set, which prevents another initialization when the program is called again.

During the startup process the user is prompted for the date. After initiation, a Main Menu is displayed, which is shown in Fig. 10 (B). The Main Menu allows the user to jump to other menus and to perform other functions, such as accessing information in files, by using the softkeys. Softkey A "Measure" jumps to the Measure Menu. Softkey B "Dacom" jumps to a menu called the Dacom Menu. Softkey C "Read Pa" allows the user to read Measure Parameters from files, and prompts for filename. Softkey D "Meas Pa" jumps to a Measure Parameter menu. Softkey E "Direct" shows a directory of connected disk drives, and the user is prompted for drive specification. Softkey F "Test M" jumps to a Test Menu. Softkey G "Store Pa" stores the current measure parameters and prompts for a filename. Softkey H "Setup M" jumps to a Setup Menu which is used to change the Assayomate setup.

The measurement Parameter menu (2 pages) is presented in Fig. 10 (C) and Fig 10 (D). The functions in the softkey portion of the menu allow values in the menu to be changed. The association of the softkey labels with the actual keys (A through H) are the same for all of the menus, and will not be repeated.

Exit jumps back to the Main Menu. Points is used to enter the number of different mixtures to be produced in a series to determine kinetic constants. The user is prompted to enter three numbers. The first is the number of points, the next is the number of repetitions to be done at each point, with the measurements to be averaged, and the third is a number of additional repetitions to be done at low substrate concentrations. Valid ranges are (2 -40), (2 - 10), and (0- 8) respectively. LRange is used to enter 3 wavelength ranges in a valid range from 180 to 820 nm.

The first is the main measure range, the second is for an additional substrate or product, and the third is an internal reference used to compensate for lamp fluctuations or electronic noise. Page 2 jumps to Page 2 of the Parameter Menu.

Conc. is for entry of the concentration of the stock solutions. The user is prompted to enter the enzyme concentration in nM and the substrate concentration in microM. Name is for entry of the name of the enzyme and the substrate in a maximum of 20 characters with no commas. [S]max. prompts to enter the substrate maximum concentration, and there is a requirement that the ratio to the stock concentration be a minimum of 4 and a maximum of 10. Epsilon prompts for four extinction coefficients for the substrate and product; 1 for each at each of two wavelengths. These are for conversion factors.

In Measure Parameter Menu 2, Spacing allows the user to select among four spacing types between concentration points for a series of assays. The choices are arithmetical, reciprocal, geometrical, and mixed. Noise prompts for two percentages: maximum noise tolerance and maximum non-linearity tolerance. Time W is for time windows. The user is prompted to enter a start and stop time at Smin, at Smax, and a measure interval and integration time. Page 1 jumps back to Page 1 of the Parameter Menu. TN Sca prompts for scale parameters for the setup of the Y-axis of plots of Turnover Number to be displayed or printed from the data collected. OD Sca prompts for input of the same sort of information for the Y-axis when Optical Density is plotted vs. time. DOD Sca prompts for similar information when difference in OD vs. time is to be plotted. Type is for entering the type of data format according to one of six implemented models. Again, for more detailed description, the Software Documentation and listings are appended, as is the complete software code for the implementation of the first preferred embodiment. (Note: The software appendices reflect implementation with three syringes. The functions coded and called are the same.)

The Setup Menu is accessed from the Main Menu by the Setup M softkey, and is shown in Fig. 10 (E). Exit jumps back to the Main Menu. Volume prompts to enter an assay volume in the range of 0.3 to 1 ml. Time is for the entry of a stopped-flow addition time of from 100 to 600 ms. Needle lets the user enter the distance the pipette must move vertically at the automatic sample changer to clear tubes supported in a rack on the stage, and the speed the drive is to travel. Syr 1 is for entry of the volume, fill level, and drive speed for the first syringe. Syr2 enters the same information for the second syringe, and syringe 3 for the third. In the event that more than 3 syringes are mounted, the menu has a softkey for each. Furthermore, the software is easily extended to the four syringe case. The mode of operation and entry is the same. Tray is for entry of certain parameters of the automatic sample changer tray, such as the speed for the drive and the number of positions in the tray for tubes (probes).

The Measure Menu is shown in Fig. 10 (F) and 10 (G), and is the menu where a series of assays may be initiated, and the results stored and printed. There are two screens to this menu. Main M jumps to the Main Menu. Page 2 jumps to the second page. Syringe jumps to the syringe control functions. Wash is used to wash a selected syringe, and the operator is prompted to enter the number of cycles and the syringe. DiVar displays the printer header so that a parameter set may be checked before a series is started. Test M jumps to the Test Menu. [E,S] allows reentry of the enzyme and substrate concentration, in case they may have changed. Points allows reentry of the points functions.

The softkey functions of the second page: Page 1 jumps to the first page. Print is a toggle that selects a different amount of printout. Content shows a directory of the data disk. Comment allows a user to enter 64 characters of comment without a comma. KmMeas starts the measure for a kinetic constant, after prompting the user for series, filename, and the enzyme concentration. Blank allows the entry of a blank file to be subtracted from all of the following measurements. [E] Auto selects between an automatic and manual mode. Single A performs a single assay.

Fig. 10 (H) is the Single Assay Submenu below the Measure Menu. The softkey functions are: Exit jumps to the Measure Menu (1 level up). Go On proceeds to the next task if in manual operating mode. Graph stops automatic operation and shows a graphic display. Alpha shows an alphanumeric display. [E] prompts the user to enter a new enzyme concentration, then mixes and measures a single assay, and proceeds with automatic operation. Activ. prompts the user to enter an activity, then tries to mix an assay with the activity that has been entered. [I] prompts the user to enter in inhibitor concentration, then mixes and measures an assay. Syringe jumps to the Syringe Control Menu.

Fig. 10 (I) shows the debug submenu. It can be invoked during automatic operation, which is then suspended, and offers several functions designed to cure problems that might develop. The softkeys are inactive for the Debug Menu. A number entry performs the function listed. Repeat Single Assay prompts the user for the assay to repeat. Next probe ready prompts for concentrations, and allows the user to build up a job queue.

After all points have been measured, control goes to the End submenu, which displays the data for all the points that have been meaasured. In the automatic mode the End Menu is active for about 10 seconds, then data is stored and the next enzyme probe is sucked in. Appendix B provides a more detailed description of the End Menu and associated softkey functions.

The Test Menu (second level) is provided to test baselines, wash syringes, and activate third-level menus. A softkey is provided in this menu to inject non-absorbing buffer to generate a zero baseline for the analytical instrument in use. Further detail is available in the Appendices to this specification.

The Test Measure submenu is a third-level menu under the Test Menu, and its function is the evaluation of assay condition for later use in the Auto Measure Menu, or for the automatic determination of kinetic constants. A typical menu and softkey functions are described in detail in the appendices. There are two pages to this menu in the first preferred embodiment.

The Auto Measure submenu is also a third-level menu under the Test Menu, and its function is the acquisition of a batch of assays with the same probe and the repetitive application of the batch measurement to a series of probes. The softkey functions and detailed description are in the appedices to this specification.

The Measure Concentration Submenu is a third-level menu for measuring the concentration of substrates and products. It is also used for measuring spectra of samples at different dilutions. Again, detail is provided in both the software documentation and listings appended to this specification.

The Dacom menu is a second-level menu for Data Communication to a host computer, and is also used for showing the header of the data curve in memory, editing the data curve, copying blank files, and plotting data. The Data Transfer Submenu under the Dacom Menu (third-level), has control functions for initiating and aborting data transfer. The softkey functions and additional detail are in the Appendices.

The Display Variable Submenu under the Dacom Menu displays variables and provides softkey functions for data editing a transfer.

The Edit Data Submenu is a third level menu under the Dacom Menu, anb provides softkey functions to peruse and edit data files before plotting or other output function.

The Plot Submenu allows the user to Title and prepare a plot before execution. There are two sets of softkey functions, and additional detail is available in the appended documents.

All of the above menus are described in more detail in Appendix B, the Michkin Software Documentation.

The Fit Software

In the first preferred embodiment, the "local" controller is a Z-80 microprocessor, and there is a limit to the sophistication and the range of software

that can be resident and operable at that level. Attachment to the HP-85 however, provides a second level for programmed control, and communication to a larger host, such as the HP-150 allows even more sophisticated post-acquisition data processing to be done. In the first preferred embodiment, the Kinetic Software described above runs on the HP-85. The Fit Software described in the present section resides on the larger host, which, in the first preferred embodiment is the
HP-150. In an alternative preferred embodiment the master processor for the microprocessor controller is an 80286, and the functions of the Kinetic Software are programmed on the 80286, while the Fit functions reside on a built-in Mac II or PS/2 computer.

The broad function of the Fit software is data analysis and display. Functions are accessed, as in the Kinetic Software, through a menu structure, and are described in detail in Appendix C, the Michfit Software Documentation. The software is written around a workfile of 100 records in the first preferred embodiment, and the records are used as operands of mathamatical operations or as sources for direct curve fits or graphical presentations. The records are those transferred by the Kinetic software typically as a result of Assayomate operations, although data from other sources may be received and processed as well.

The Menus include:

Input Menu, which is responsible for all inputs from outside of the workfile.

Output Menu, which is responsible for sending data to several output devices.

Data Handling Menu provides functions to edit the workfile, to perform operations with registers, and to exchange the active workfile with stored workfiles.

Interpreter/Macros provides an operations interpreter and a macros function for automating sets of procedures that have to be done regularly.

Direct Curve Fit provides a new fitting program for the Fit Software. It is started, as are other direct curve fit routines, by entering first estimates. It may also use a grid search technique covering the entire meaningful range of all parameters. A binary gradient search routine is used to optimize the first estimate found by the grid search. There are several models built into the software, such as (1) Michaelis - Menten, (2) Consecutive Reaction, (3) 2 Km 2 Vmax values, (4) Hill Equation, (5) Non-cooperative sites, (6) Sequential interaction, (7) Competitive Inhibition, (8) Uncompetitive Inhibition, (9) Non-competitive Inhibition.

There is also an equation interpretor so that a user
can enter his own model.

Plotsize Menu has the major function of the graphical presentation of data on a screen, plotter, or printer.

Display Menu provides a comprehensive set of functions to create and annotate displays of data.

The Fit Software Documentation and software listings are Appended to this specification, and provides detail of all menus and functions.

Examples of Use

A principle use of the apparatus of the first preferred embodiment is the determination of kinetic constants for enzymes by measuring large numbers of assays quickly, accurately, and efficiently, as has been described above. This is, however, not the only application of importance. The apparatus as described may be employed as well for screening operations. For example, by stocking one reservoir with a particular substrate, or even more than one reservoir with more than one substrate, and providing a plurality of enzymes in probes at the automatic sample changer, which may be constituted to have a large number of probe positions, a user may investigate, quickly and automatically, the specificity of enzymes for the substrates. Many similar screening investigations may be arranged and performed with the apparatus and software, such as the effects of a plurality of inhibitors or accelerators on particular reactions.

By use of the manual mode, driving each of the functions of the apparatus with keystroke commands through the functions listed in Appendix A, and also by tailoring software to drive the apparatus through the same functions via the communications bus, a truly broad range of experimental and analytical procedures may be accomplished. Screening operations can be performed, simple kinetic constants can be determined, inhibitors, activators and accelerators can be tested, the effects of variable ionic strength on reactions can be quantified, and dilutions can be performed, among many other variations in technique.

A major advantage of the invention is that dilutions may be performed to prepare solutions over a very broad range of concentrations, which has not heretofore been possible automatically. For example, if line 169 (Fig. 5) extends to a reservoir containing a stock enzyme solution, one might draw from the enzyme reservoir with valve 151 activated and valve 167 deactivated, to an extent of one one-thousandth of the volume of syringe 153 (48 motor pulses in the Assayomate), then switch valve 167 to fill the syringe from a probe on the automatic sample changer with buffer (the balance of 48,000 steps). The result will be an accurate dilution of the enzyme solution by a factor of 1000 to 1. The buffer could be placed in an empty probe, if desired, from another stock reservoir by using another syringe and its associated valves. Moreover, if desired, the dilution could be compounded by ejecting the 1000 to 1 diluted solution to another empty probe on the stage of the automatic sample changer, and the 1000 to 1 dilution could be performed again, drawing the originally diluted solution in place of the stock enzyme. The result would be a solution accurately diluted by a factor of 1,000,000 to 1. Depending on specific needs, the same dilution could be done in several different ways than described in this particular example.

With a large number of probe positions on the rotary stage at the automatic sample changer, and the use of the several syringes with fine gradient control and the valve block for switching solutions to a variety of different paths, a truly flexible analytical instrument is made available. There are, for example, many procedures that are commonly required to activate enzymes, preincubate inhibitors, and do other sorts of preprocessing, before the particular reactants are useful in preparing assays for determining kinetic constants. The apparatus of the invention, together with software written for the purpose, allows such preprocessing to be done automatically. Software for this purpose is attached as Appendix H through Appendix L. One particular aspect of this automatic preprocessing that is important is automatic dilution. Once an initial measurement is made, if the concentration of one of the reagents is too high, the system will automatically dilute the reagent until it is brought into a useful range. With this feature, the apparatus becomes truly automatic.

Example

Figs. 11-12 show the results of a determination of the Mikaelis constant for cytochrome c. These figures show the printouts provided by the Assayomate as a result of that determination. In the printout shown in Fig. 11, a header is shown in the upper portion which indicates the file name, the name of the disc volume, and the date of the measurement. Below that the measure parameters are printed. Line 1 indicates the type of measurement (Michaelis constant initial velocity). Lines 2-4 indicate time window at lowest and highest substrate concentration. Line 5 indicates measuring interval and integration time for each measurment. Line 6 indicates noise, and determines the standard deviation within which single measurements must fall so that the average is calculated. Lines 7-9 indicate the two monitoring wavelength ranges and wavelength range for normalization. Lines 12-13 indicate concentration of enzyme in stock and in assay. Lines 12-13 indicate concentration of substrate in stock and highest substrate concentration in assay. Line 14 indicates the number of different substrate concentration (points) and number of repetitions for single substrate concentration. Lines 15-16 indicate extinction coefficient for wavelenght ranges on line 7-8. Line 17 indicates volume of assay (sum of buffer, enzyme and substrate). Fig. 12 shows a printout of the change in absorption at 548-552 nm (DELTA OD/sec) in the first column and standard deviation in the third column for three determinations. In the second column the decrease of absorption at 416-420 nm is shown. Both ranges are normalized to the absorption at 580-586 nm. In Fig. 13 is shown a plot of the derivative of absorption (DELTA OD/sec) as a function of time at 20 different substrate concentrations. Fig. 14 is a printout showing the turnover numbers at 20 different substrate concentrations calculated from data of Fig. 12 with the difference extinction coefficients and the enzyme concentration given in Fig. 11. Shown in the first column is the point number. The second column is the turnover number determined from the absorption change at 548 to 552 nm. In the third column is turnover number determined from th eabsorption change at 416 to 420 nm, and in the fourth column are the substrate concentrations in uM. Fig. 15 shows a plot of turnover number versus substrate concentration using the values from Fig. 14. Also indicated is the maximum velocity (or turnover number) in 1/sec and Michaelis constant in uM calculated from linear regresion of double reciprocal transformation of data from Fig. 14. The correlation coefficient of linear regression is also shown.

Many other determinations have also been made with the Assayomate with excellent results. For example, studies have been made of alcohol dehydrogenase and aspartate aminotransferase, both of which have been extensively studied by others, and for which kinetic constants have been widely published. Measurements of the kinetic constants obtained by the Assayomate were well within the bandwidth of the published constants. Manual determination of kinetic constants with the same lot of enzyme using the same buffer conditions resulted in the same kinetic constants. Reproducibility of the results was much better for the Assayomate than for manual determinations, however. Fig. 16 shows the results of a reproducibility analysis for cytochrome c.

It will be apparent to those skilled in the art that there are a variety of changes that may be made, deviating from the above descriptions of the preferred embodiments of the invention, without departing from the spirit and scope of the invention. Foe example, the computers described are not the only suitable machines that might be incorporated. There are many others, and software might be altered to accomodate substitutions of computer equipment. The microprocessors, communication protocols, data structures, and other details were incorporated as matters of compatibility and convenience, as well, and there are competent substitutions that might be made.

There are, in the structure of the apparatus, many changes and substitutions that might be accomplished without departing significantly from the spirit and scope of the invention. The number of driven syringes can vary, as can the number of stock reservoirs and the number of probe positions on the automatic sample changer stage. The stage itself need not be a rotary device, as there are a number of other ways that a plurality of probes might be presented to a pipette. There could be more than one automatic stage and pipette, too. The variety of such possible substitutions is quite large, and not all may be listed here; but these sorts of alterations do not depart from the spirit and scope of the invention.

Appendix A

Syringe Control Menu (menu prefix *)

A ALTM	This function changes the volume that is moved by the functions B (give out), C (suck in), and D (give back) of the active syringe. The message displayed is 'Please enter the volume in (ML)', it expects the input of a number between 0 and the maximal volume.
B GIVE	This function moves the volume defined by function A toward the mixing chamber and the measure head immediately. It displays 'GIVE OUT' during the operation.
C SUCK	This function sucks the volume defined by function A from the reservoir into the syringe immediately. It displays 'SUCK IN' during the operation.
D BACK	This function moves the volume defined by function A back to the reservoir immediately. It displays 'GIVE BACK' during the operation.
E WASH	This function washes the active syringes by repetative suck in and give out cycles. It displays "PLEASE ENTER THE NUMBER OF CYCLES' and expects the input of a positive number in the range from 0 to 99. The first cycle empties the syringe, the next cycle sucks in 2% of the maximal volume, the third empties the syringe and so on. An odd number of cycles, stops washing with an empty syringe,

	an even number stops washing with a filled syringe. The message 'WASHING OF ACTIVE SYRINGE' is displayed during the entire operation.
F EMPTY	This function empties the syringe back to the reservoir immediately. It displays the message 'EMPTYING SYRINGE' during the operation.
G FILL	This function fills the syringe from the reservoir to the level entered by the function '=' immediately, it displays the message 'FILLING SYRINGE' during the operation. If the fil level of the syringe is already higher than the maximal fill level, the command is ignored.
H STOPFL	This function mixes an assay and displays the message 'STOPPED FLOW'. The amounts of buffer, enzyme and substrate have to be defined by the function 'K' before this function is executed. If one of the syringes runs out of volume, the function is immediately aborted. Approximately 100 milliseconds before the assay is finished, the busy bit in t he status byte of the IEC interface is cleared, and after the termination of the assay, the menu is updated.

(the following softkeys are acessed in the shift state):

I SPEED	This function is used to change the speed of piston movement of the active syringe. It displays 'PLEASE ENTER THE SPEED IN ML/MN' and expects the input of one number in the range of 0 to 99 ml/min.
J SETSF	This function is used to change the composition of the assay. It displays 'PLEASE ENTER THREE VOLUMES (ML)' and expects three parameters, the volume of buffer (in ml), the volume of enzyme (in ml), and the volume of substrate (in ml) e.g. 0.300,0.050,0.050 (cr). This function does not execute the assay, this done by function 'H'.
K FUVOL	This function furnishes the volume of a series of assays to the display of the IEC listener depending on the source of the command. It displays 'VOLUMES OF SUBSTRATE' and then according to the setting of the number of points variable n times a volume e.g. 7 0.367(cr)(lf). The listener has to receive n lines of information.
L SETRE	This function is used to set the type of substrate spacing, the number of points, the number of repetitions, the volume of the assay (in ml), the amount of enzyme (in ml), and the maximal amount of substrate (in ml) e.g. *M 1,20,3,0.400,0.040,0.100.
M FUBED	This function calculates the need of buffer,

enzyme, and substrate according to the settings of function 'M'. Then it executes a fill command for all syringes, but only the syringes containing not enough volume are actually moved. If a syringe has to be moved, it displays 'FILLING SYRINGE'.

N DILUTE ....

O LEVEL This function prompts 'PLEASE ENTER FILL LEVEL IN ML' and expects one number in the range from 0 to the total volume of the active syringe.

P SFSET This function is used to set the volumes for an assay by specifying which point of the active rowtype and number of points shall be prepared. Before this function can be executed, proper parameters for function 'M' have to be entered. It prompts 'PLEASE ENTER POINT (ASSAY)' and expects an integer in the range from 0 to the maximal number of points.

Other functions : (valid in all menus)

R SWREM Switches the ASSAYOMATE to remote control, after this function has been executed commands from the keyboard are no longer accepted. The only exception are the screen roll and the LOCAL function. It displays 'SWITCHED TO REMOTE'.

S DIHEME This function prompts "PLEASE ENTER

HEXADECIMAL ADDRESS' and expects a hexadecimal number in the range of 0 to FFFF. It displays 8 bytes of memory in hexadecimal starting with the memory location that was entered. *S F800

T LDHEME	This function prompts 'PLEASE ENTER HEXADECIMAL ADDRESS' and expects a hexadecimal number in the range of 0 to FFFF. The hexadecimal address can be followed by to 32 pairs of hexadecimal numbers each separated by a space from the next pair. This function converts the pairs of hexadecimal numbers to binary (0-255) and stores them to the ASSAYOMATE memory. *T F800 AA CF 01 (cr)
U LOCAL	Switches to local (commands from keyboard accepted). It displays 'SWITCHED TO LOCAL'.
V RECASC	This function expects a hexadecimal address and a string of ASCII characters from the IEC talker. The ASCII string is stored to the ASSAYOMATE memory.
W SEASC	This function epects a hexadecimal address and sends the content of the ASSAYOMATE memory to the IEC listener until a carriage return is encountered. The longest string that will be sent is 128 bytes. If the IEC listener is not ready to receive data, the ASSAYOMATE displays 'IEC TIMEOUT' after 1 second and aborts the operation.

X MSTAT	This function reads the status of the microswitches and copies it to the status byte.
Z NUWFU	This command executes the RAM based routine that has been loaded to the ASSAYOMATE memory before.
O AKTO	This command deactivates all syringes.
1 AKTA	Keys 1, 2, 3 and 4 activates the buffer, the enzyme, and the substrate syringe, respectively.
* SYRCO	This command activates the syringe control menu if it is not already activated.
+ SAMPC	This command activates the sampler control menu if it is not already activated.
- DETCO	This command activates the detector control menu if it is not already activated.
/ MICHKIN	This command starts or restarts the kinetic software. If the kinetic software is started for the first time after power on, ...

APPENDIX B
The Michkin Software Documentation
By Bruno Michel

chapter 1

Introduction

The MICHKIN user software is the interface between the user on one hand and the ASSAYOMATE and the photometer (or another detector) on the other hand. It was written to ease the acquisition of kinetic constants as much as possible. Six menues offer a well arranged set of commands which are needed by the user.
The determination of a kinetic constant is executed by simply pressing a softkey in the Measure Menu.
Measurement parameters are edited in the Parameter Menu. They can be stored on disc to be used at a later time.
During the data acquisition for a kinetic constant, the MICHKIN software calculates the amount of buffer, enzyme, and substrate of inhibitor for the data point i and sends the command to the ASSAYOMATE (functions *O and *H). When the mixture is finished, the spectrophotometer is activated and the MICHKIN software accepts series of spectra from the spectrophotometer. From the spectral data the software calculates a normalized time dependence of the optical density at one or two wavelength ranges. It then differentiates the data and checks the stability and the noise of the derivative.
Each concentration is repeated several times (defined in the Parameter Menu) so that a standard deviation can be calculated and checked. After all data points have been measured successfully, the data is stored on disc and the kinetic constants are determined and printed (see Sample Result).

chapter 2
System concept

The computer controllable ASSAYOMATE and a commercial photometer (absorption / fluorescence) are used by the MICHFIT software to determine kinetic constants from a set of steady-state activities measured under different conditions.

2.1 The MICHKIN Overlays

The MICHKIN software is divided into four overlays, a start overlay and three overlays, that can be activated from the start overlay. Data is exchanged between the different overlays with the aid of a common data block. The MICHKIN software has been written in HP BASIC
All overlays are oganized in a similar manner; they implement a second level menu:

Measure overlays	: Measure Menu
Testmeasure overlay	: Test Menu
Data Transfer overlay	: Data Transfer Menu

From the second level menus it is possible to branch to lower level menus. In all the lower level menus softkey #1 exits from any place to the second level menu of the respective overlay. Pressing softkey #1 in one of the second level menus exits to the Main Menu, this is the highest level (first level) menu from where all operations start.

2.1.1 Start Overlay ('Autost')

- Starts MICHKIN software package
- Initializes the COMMON data block
- Implements Main Menu (1st level)
- Stores measurement parameters in files
- Reads files containing measurement parameters

- Displays directories of all discs
- Purges data files and initializes data discs
- Activates overlays
- Implements Parameter Menu (2nd level) for input of syringe parameters (e.g. L-ranges, duration, concentrations)
- Implements Setup Menu (2nd level) for input of syringe parameter (e.g. fill speed, fill-level, assay volume)

2.1.2 Measure Overlays

- Implements Measure Menu (2nd level menu)
- Measures single assays (input of enzyme and substrate concentration)
- Changes enzyme and substrate concentrations
- Alters number of single measurements (points) and number of repetitions of points
- Measures blanks
- Measures Michaelis constants (raw data are stored on disc)
- Subtracts blanks from Michaelis constant measurements
- Measures series of enzyme probes with autosampler
- Implements Syringe Control Menu (3rd level menu)
- Controls the syringes (fill/empty/suck in/give out/give back)

The main functions of the measure overlays are acquisition of photometer data, storage of results and calculation to the kinetic parameters Km and Vmax. For this purpose the overlay has to be able to control both the photometer and the ASSAYOMATE. The Syringe Control Menu is used to move the syringes under manual control. This menu offers the functions fill, empty, suck in, give out, give back and change amount, described in the

ASSAYOMATE Documentation and in the MICHKIN User Manual. The syringes are activated one after the other by pressing softkey #8 in this menu.

An assay is mixed and measured by activating the single assay function in the Measure Menu. After the softkey <Single Assay> has been pressed, the user is asked to enter the substrate and the enzyme concentrations. If the entered concentrations are valid, the assay is mixed and the photometer starts measuring. The optical density is plotted on the screen as a function of time, then the derivative, the linearity and the noise are printed.

Starting from the same menu, the Michaelis constant together with the maximal velocity is determined by pressing the <Km meas> softkey in the Michaelis overlay. The user has to enter the name of the data file e.g. 'TEST A1', and the enzyme concentration of the stock solution. Then a single assay is mixed and the measured turnover is compared with the selected activity. Depending on the result of this test, the amount of enzyme in the assay is increased or decreased and the assay is repeated. If the turnover number is in the correct range, the header is printed and assays with different substrate concentrations are mixed and measured. After the last of these assays , the program proceeds to the End Menu. In this menu one or several averages (of assays) can be repeated or the result can be stored immediately. Then the kinetic parameters are calculated from a linear regression in a double reciprocal transformation and printed together with the correlation coefficient.

2.1.3 Testmeasure Overlay

- Implements Test Menu (2nd level)
- Tests assay conditions

- Tests series of enzymes (velocity) with autosampler
- Measures subtrate and product concentrations (if they absorb UV/vis light)
- Measures spectra of substrate and product

The test overlay has three major functions. In the Concentration Menu the concentration and spectra of substrate and product can be measured. The Testmeasure Menu is used to test assay conditions for new substrates or enzymes. In the Serial Assay Menu, a set of assays can be performed on a large number of probes, either manually or automatically by means of an autosampler.

2.1.4 Data Transfer Overlay

- Implements DACOM Menu (2nd level)
- Transfers files to host computer via serial interface
- Averages blank measurements
- Copies data files
- Edits data files
- Plots data (turnover number vs. substrate, double reciprocal, Eadie Hofstee or Hanes Wolf transformation)

The main function of this overlay is the transmission of data via RS-232 interface to a host computer. Data files that have been stored on disc are read in and sent to the host. The data transfer is controlled by this overlay if the softkey 'Master' is selected in the Data Transfer Menu. The data transfer has to be controlled by the host computer if the softkey 'Slave' is selected. In addition to data transfer, this overlay is responsible for editing of result files, and plotting of data on an external plotter.

2.2 Sample Result

Before the photometer, the computer and the ASSAYOMATE are switched on, they have to be connected by the IEC bus.The photometer and the ASAYOMATE are then switched on, they perform the initiation simultaneously.

Starting from the Main Menu, a measurement parameter file has to be read in. Then the concentration of the substrate is determined with the Testmeasure overlay. When the Testmeasure overlay is executed for the first time, syringe parameters are loaded into the ASSAYOMATE memory. But before the photometer is able to perform the first measurement, a 'Reference Measurement' has to be performed. To do that, buffer is sucked into syringe 1 (either with functions from ASSAYOMATE keyboard or with the Syringe Control Menu). The reference measurement can be executed from the Test Menu (see MICHKIN User Manual). After that the substrate concentration is measured in the Concentration Menu. Then the concentration of the enzyme is determined with the Testmeasure Menu.

Printouts of result from the determination of kinetic constants can be generated in two modes: If the print flag is on, a full printout is generated a) - h), otherwise only a) and h) are printed.

A typical full length printout consists of 8 parts.

a)	line 1	Header with serial number of experiment (B5) and file name (HORSCC)
	line 2	Date, disc volume name (Modcyt)
b)	This is a list of important measuring parameters	
	line 1	Type of measurement (Michaelis constant initial velocity)
	line 2 - 4	Time window at lowest and highest

	substrate concentration
line 5	Measuring interval and integration time for each measurement (integration time $\leq$ measuring interval)
line 6	Noise, determines the standard deviation within which single measurements must fall so that average is calculated.
line 7 - 9	Two monitoring wavelength ranges and wavelength range for normalization (no absorption change during reaction in this range)
line 10 - 11	Concentration of enzyme in stock and in assay
line 12 - 13	Concentration of substrate in stock and highest substrate concentration in assay
line 14	Number of different substrate concentrations (points) and number of repetitions for single substrate concentration.
line 15 - 16	Extinction coefficient for wavelength ranges on line 7-8. Negative sign indicates decreasing absorption during reaction.
line 17	Volume of assay (sum of buffer, enzyme and substrate)

c) The change of optical density per second is printed for both wavelength ranges. The third column lists the absolute standard deviations of the averaged points of range 1.

d) Plot showing the derivative of the time course of the optical density versus time (deltaOD/sec.) for all data points. Horizontal straight lines in the ideal case. This plot gives a visual impression of the time independence of the initial velocities.

e) Header a) is repeated and two lines of comment are printed.

f) Table of the turnover numbers at different substrate concentrations used for the generation of plot g).

h) Result of a linear regression of kinetic data in a double reciprocal transformation.

The accuracy of kinetic constants is increased when the measurements are corrected for a blank reaction. Such a blank can be measured by setting the enzyme stock concentration to zero. It is then stored as deltaOD/sec. table on the system disc and subtracted from all subsequent measurements.

Chapter 3
Data Structures

3.1 Common Data Area

The common data area of the MICHKIN software is a portion of RAM which is reserved in all overlays. It is used to exchange spectra, measurement parameters, flags, and pointers between the overlays. The entries in this block are:

S1(80)	time course 1 (absorption versus time data)
S2(80)	time course 2 (absorption versus time data)
S3(80)	average of 1 (absorption versus time data)
S4(10)	scratch buffer
T1(41)	titration curve 1 (deltaAU / s)
T2(41)	titration curve 2 (deltaAU / s)

D1$[1]	*	series of measurements
D2	*	number of measurement
D4$[20]	*	flags (blank correction)
E1$[20]	*	name of enzyme
E2$[20]	*	name of ligand
T1$[65]	*	comment (64 characters)
T(20)	*	measurement parameters
L(20)	*	measurement parameters
B(20)	*	measurement parameters
W(20)	*	syringe set up (volumes, speed)
W1(20)	*	stopped flow commands
A(5,5)	*	axes of plots
D3$[20]		date of measurement
C(15)		counters of syringes (in ml)
E(20)		enzyme concentrations in autosampler (uM)
Z5-Z9		flags
B9$[8]		name of volume (disc)

(*) stored in measurment parameter file

3.2 Measurement Parameter File
The measurement parameter files are generated after the 'Sto Pa' softkey has been pressed in the Main Menu of the MICHKIN software. They are used to store the set up, conversion factors, and enzyme names. Such a file is generated for each different enzyme or substrate.
The meaning of the entries is:

T(1)	start wavelength of curve 1
T(2)	stop wavelength of curve 1
T(3)	enzyme conc. in nM
T(4)	substrate conc. in uM
T(5)	no. of points
T(6)	start time of time window at max.

	substrate
T(7)	stop time of time window at max. substrate
T(8)	number of repetitions
T(9)	data format
T(10)	start wavelength for internal
T(11)	stop wavelength reference
T(12)	start wavelength range 2
T(13)	stop wavelength range 2
T(14)	[S] max. (uM) actual
T(15)	extinction coefficient substrate range 1
T(16)	extinction coefficient substrate range 2
T(17)	difference extinction coefficient range 1
T(18)	difference extinction coefficient range 2
T(19)	row type
T(20)	additional repetitions
B(1)	maximal noise in %
B(2)	maximal nonlinearity in %
B(3)	actual enzyme conc. in assay
B(4)	starting enzyme conc. (autor)
B(5)	preselect enzyme in assay
B(6)	actual substrate conc. in assay
B(7)	assay volume in ml
B(8)	Twart
B(9)	delta Twart
B(10)	activity [E] auto
B(11)	measurement interval (s)
B(12)	integration time (s)
B(13)	t-start1 of time window at

B(14)	t-stop min. substrate conc.
B(15)	-
B(16)	-
B(17)	grad E
B(18)	dilution factor
B(19)	tube counter
B(20)	washing syringe

3.3 Format of Transmitted Data
The MICHKIN software measures and stores velocities of enzyme reactions at different substrate concentrations (titration curves). These results are sent to the host preceded by a header (measurement parameters). The format of the transmitted data is:
1st. line: header
A..1date.......flags.......enzyme name...........
substrate name ....comment (64 characters).

2nd line 20 variables
format: SDDDDD.DDDDD

3rd and fourth line optional variables with same format

data points in floating point format:
velocity 1, verlocity 2, substrate conc.
SD.DDDDEEEEE SD.DDDDEEEEE SD.DDDDEEEEE

3.3.1 Configuration of Serial Interface
9600 baud
8 bit 2 stop bits
no parity
Xon/Xoff handshake

Chapter 4
Description of Menus and Algorhytms

4.1 Start Overlay Autost.KSYS

4.1.1 Initialization

When the start overlay is activated it defines common and global variables (lines 120 to 220, see data structures for more information). The initialization flag is checked if it is set to 1 (C(8) = 1). If this is the case, the Main Menu is shown (line 4000). When the MICHKIN software is started, the variables in the common data area are still undefined. Reading such a variable generates an error. This is used to test if initialization is necessary. During the initialization a copyright screen is shwon (subroutine on line 165-290). All variables are set to default values on lines 295-398. The software prompts for the date and for the name of the data diskette (D3$, B9$, lines 400-402). The content of the file 'INHALT' is shown by the procedure on line 2000. This file contains 12 lines of information about the content of the data files on this diskette. A special parameter file 'DEFAULT' is then read by the READ_PARAMETER procedure (lines 4410ff). If this has been completed successfully, the Main Menu is shown (line 4000).

In case of an error during reading of the 'INHALT' file the small menu on lines 410-460 is shown. With this menu it is possible to retry reading (H2 = 1), rename the diskette (H2 = 2), create a new 'INHALT' file (H2 = 3), initialize a diskette (Formatting H2 = 4), or to show the directory (H2 = 5). If an error occurs while the directory is shown or while the volume is renamed, the error message *'Please check the drive'* is shown (line 450). This happens when the disc is not formatted. If the drive is not switched on, or if the diskette is damaged. After one function has been completed successfully, the Main Menu is shown.

4.1.2 Main Menu

The Main Menu shows the copyright message screen (subroutine on line 265) and adds:
'Please select as softkey:' and
'All other keys stop the program'
8 softkeys are defined by the ON KEY#X, '<Label>' GOTO YYYY (lines 4150-4220). The softkey are then shown on the last two lines of the screen with the KEY LABEL command. Line 4230 forms an indefinite loop that can only be left by pressing one of the softkeys. If, however, another key is pressed the program stops.
With the ON KEY ....GOTO commands of the HP85 BASIC it is difficult to write a structured code. GOTO jumps are avoided if possible, if they are necessary they are kept as small as possible.
If softkey #1 is pressed, the routine from line 4250 to 4317 checks if all parameters are in a valid range (procedure on line 4600) and stores the axes and labels of the three plot types in three files (procedures on lines 5000, 5100, and 5200). Two hidden temporary files are generated on lines 4290 and 4295. All spectrophotometer data has to be erased before the KINX overlays can be CHAINed. Depending on the data type parameter(T(9)) different measure overlays are loaded ('KIN1', 'KIN2', 'KININ', 'KINSU'). All these overlays contain the same Measure Menu and the same data acquisition routine. The only difference is the evaluation of the data (Km, KI, Km1+Km2) and the starting conditions (Eautom, IAutom , S substr). If the ASSAYOMATE has not yet been initialized (C(15) = 0) the

ASINIT overlay is called else the measure overlay is CHAINed.
Pressing softkey #2 displays *'The datatransfer program is loaded'* and loads the data transfer overlay from the system disc (CHAIN 'DACOM.KSYS', (lines 4320 and 4330).
Pressing softkey #3 shows the directory of the system disc (CLEAR @ CAT '.KSYS') and prompts for the file name of a parameter file (line 4405). The file is opened and the parameters are read (lines 4410-4430).
If an error occurs because the file is too short, the message *'Old file: Please update syringe parameters'* is shown. The files continuing the information for the plots are updated (line 4440) before the routine ends with the GOTO 4000 statement.
Softkey #4 activates the Parameter Menu (GOTO 500).
Pressing softkey #5 prompts for a drive specification =':D700' 1=':D701' 4=':D710' and shows the directory of the selected drive. Files can be erased (PURGE), spaces left in the directory from earased files can be removed (PACK), all files can be erased (INIT), or the routine can be left without any action (N or no input) (lines 1600-1640). If INIT has been selected the user has to confirm his intention to erase all files (line 1657). The diskette is then initialized (formatted) using the name provided by the user on line 1660, and the drive specification (H1).The INHALT file has to be reentered using the EDIT_INHALT procedure on line 2200.

Pressing softkey #6 checks if all parameters are in a valid range, updates the stored axes of the three plots, checks if the ASSAYOMATE has been initialized and loads the testmeasure overlay (CHAIN 'KONZMSG.KSYS', (lines 4340-4385). This sequence is similar to the sequence for softkey #1.

Pressing softkey #7 shows the directory of the data disc and prompts for a file name D$ (line 4505). The active syringe and measure parameters are stored in a file on the systemdisc:

CREATE D$,5,256	This command opens a file with 5 records
ASSIGW#1 to D$	The file buffer is opened
PRINT#1;.....	The data is written to the buffer
ASSIGN#1 to *	The file buffer is flushed and the file is closed

The sequence ends with a GOTO 4000 statement. Pressing softkey #8 activates the Setup Menu (or Syringe Parameter Menu) (GOTO 1000).

4.1.3 Parameter Menu

Lines 500 to 540 are used to generate the following screen:

```
 *****  Measure Parameter Menu 1  *****
 Enzyme name                     40    nm (E)
 Substrate name                   250 um (S)
 max conc. in assay                30 um
 Range 1   from     548 to     552  nm
 Int. Ref. from     580 to     586  nm
 Epsilon    Range1:   to     Range 2:
 Substrate:    25.2      121.1    1/(mM cm)
 Product  :     9          78    1/(mM cm)
 No. of Points        20
 No. of Repetitions   3         3
  Conc.       Names       [S]max     Epsilon
  Exit        Points      LRange     Page 2
```

The softkeys are defined on lines 550-564 and shown on the screen using the KEYLABEL command on line 566. The indefinite loop can only be left when a softkey is pressed.

Softkey #1 exits to the Main Menu (GOTO 4000)

Softkey #2 is used to input the number of repetitions T(8) and the additional repetitions T(20) (lines 840). The input is rejected if the total number of data points is larger than 100 (T(5) * T(8) > 100) if the number of repetitions is larger than 10 (T(8) > 10) or smaller than 2 (T(8) < 2). The sequence ends with the GOTO 500 statement.

Softkey #3 is used to edit the wavelength ranges for the data acquisition (lines 760-820).

Range 1:	T(1), T(2)
Range 2:	T(12), T(13)
Normalizing:	T(10), T(11)

The validity of the inputs is checked in the CHECK.DIODES procedure on line 4800.

Softkey #4 activates the second screen of the Parameter Menu.

Softkey #5 is used to edit the values of the enzyme concentration (in uM) (lines 880-885).

Softkey #6 is used to edit the names of enzyme (E1$) and substrate/inhibitor (E2$, lines 690 and 695). The strings are input in the typewriter mode (shift --> CAPITAL). The HP 85 switches this mode on and off with the FLIP command.

Softkev #7 sets the maximal substrate concentration [S max] (T(14)) in micromolar (lines 890-910). The input is rejected if the maximal substrate concentration is lower than one fifteenth of the stock concentration or if it is higher than a fourth of the stock concentration (T(14) < T(4) /15 or T(14) > T(4) / 4). Then TN vs. [S] plot is adjusted to [S]max by means of the AXES_TNS procedure.

Softkey #8 is used to reenter the extinction coefficients for substrate and product for both ranges (lines 700-706). The extinction coefficient for the substrate T(15) and difference extination coefficient (product minus substrate: T(17)). The same applies to the second range (T(16), T(18)).

The second screen of the Parameter Menu is generated on lines 600-640. Again a set of 8 softkey labels is defined and shown on the last two lines by means of lines 670 to 686:

Screen 2 of Parameter Menu:

```
*****  Measure Parameter Menu 2  *****
Measure type :  Michaelis in velocity
Time Window :   Mixed row
at [S]max.      3.2  to  8    sec
at [S]min.      1.2  to  2.4  sec
Max. Noise Avg. 3% Max. Nonlin. 15%
Y-Axis TN vs. [S]-Plot :
   0     100       20       0
```

```
Y-Axis delta OD vs. Time Plot
  -0.01    0.001    0.005   0
Y-Axis OD vs. Time Plot :
  -0.1     1.5      0.2     0.1
TN Sca     OD Sca   DOD Sca   Type
Spacing    Noise    Time W.   Page 1
```

Softkey #1	selects among several types of data spacing (lines 995-997). 1 geometrical spacing 2 arithmetical spacing 3 mixed spacing 4 reciprocal spacing The data spacing variable T(19) is incremented and set to 1 if it is larger than four. The current spacing type is shown in the menu (lines 620-626) depending on the value of T(19).
Softkey #2	is used to modify the selection criteria for the kinetic data acquisition a) the linearity b) the reproducibility of assays B(2) The variables are checked if they are in the range from one to one hundred percent (lines 920-930).
Softkey #3	is used to set the duration of the assays (lines 740-755). The duration at [S]max (start T(6), stop T(7)) is different from the duration at [S]min (start B(13), stop B(14)). Here the measure interval (B(12) < 2) and the integration time (B(11) < 1) can be set. Intervals and integration time are only accepted if the interval is larger than

the integration time and if they are truncated to one digit in the fractional part. The time windows are further checked if the stop time is larger than the start time and if the number of data points (T(7)/ <2) is not larger than 80. The stop time at [S]min should not be larger than the stop time at [S]max (lines 745-750). The axes for the OD plot and the delta OD plot are adjusted in the AXES_OD and AXES_DELOD procedures.

Softkey #4 jumps back to the first page of the Parameter Menu (GOTO 500).

Softkey #5, #6, and #7 are used to modify the Y-axes of the TN vs. plot, the OD vs. time plot and the delta OD vs. time plot. The program sequences can be found on lines 960, 940, and 980. The axes definitions minimum A(,1), maximum A(,2), increment A(,3) and first increment A(,4) are input and checked. Then the axes are updated with the appropriate procedure.

4.1.4 The Setup Menu

The screen of the Setup Menu (or Syringe Parameter Menu) is generated using the code from lines 1000 to 1080.

```
**********   Setup Menu   **********
          Volume   Fillevel   Speed
           (ml)      (ml)   (ml/min.)
```

```
Syringe 1    50.0     50.0     45.0
Syringe 2     5.0      5.0      4.0
Syringe 3     2.5      2.5      2.0

Tray        100.0    100.0     50.0
Needle        1.0      1.0     10.0

Stopped flow time   600.00 ms
Volume per assay      0.50 ml
Syr 1    Syr 2      Syr 3    Tray
Exit     Volume     Time     Needle
```

A set of 8 softkeys is provided to modify the parameters in this menu:
No checks for the validity of the parameters are performed in this menu. (For lack of RAM memory). The user has to know the valid ranges of the parameters.

Softkey #1 exits to the Main Menu (GOTO 4000).

Softkey #2 is used to enter the stopped flow parameters

W1(1)	volume of buffer
W1(2)	volume of enzyme
W1(3)	volume of substrate

These parameters are used to determine the assay volume: B(7) = W1(2) + W1(3). Valid ranges for buffer 0 - 1 ml, for enzyme 0 - 0,2 ml, and for substrate 0 - 0,2 ml. All amounts have to be given in ml. Valid range for assay volume: 0,4 - 1 ml.

Softkey #3 is used to enter the acceleration ramp angle for stopped flow operations W1(5) and the speed of the stopped flow addition W1(4). The speed is inverse proportional to the cycles of the primary routine (see OPERATING SYSTEM) 65536 cycles =~ 0.5 seconds 32'000 cycles ~ 0.25 seconds a.s.o.. Selecting two short stopped flow times reduce the accuracy of the added amounts and reduce the variability of substrate mixing. In order to achieve a wider range of substrate for example stopped flow times are increased for the highest substrate amounts (see OPERATING SYSTEM).

Softkeys #4, #5, #6, #7, and #8 are used to enter the parameters for the syringes.

1	W1(1)	Volume (in ml) moved by syringe with 60 mm piston movement.
2	W1(6)	Maximal fill pointer: The syringe is moved to this position (in ml) if a FILL command is given.
3	W1(11	Pointer for relative movements: The amount (in ml) stored in this variable is moved when a SUCK IN, GIVE OUT or GIVE BACK command is given.
4	W1(16)	Speed of movement in (ml/minute)

The table above is given for the first syringe (Softkey #5) the variables for the second syringe belong to the same array but with each index increased by one. The same applies for syringe 3. The fourth syringe or the tray is controlled in the same way. The fifth syringe or needle is controlled using the same array. Tray and needle are the labels because the stepper motors can be used to move a tray and a needle of an autosampler.

4.1.5 Start Overlay Procedures

DISPLAY_INHALT procedure (lines 2000ff)

This procedure reads the file INHALT on the diskette with the name B9$. It clears the screen and displays a header, then it opens the file (lines 2025 and 2030). It reads and displays ASCII strings until the EOF mark generates an error (lines 2040-2060). The file is closed and the procedure ends.

EDIT_INHALT procedure (lines 2200ff)

The file INHALT is erased and a new file with the same name is opened. It is opened and the prompt *'Please enter max. 12 lines of comment'* is displayed. Up to 12 lines with a max. length of 32 bytes are accepted (lines 2250 and 2255). If the counter reaches 12 or a string with length 0 is encountered, the file is closed and the routine is left.

CHECK_PARAMETER procedure (lines 4700ff)

The measure ranges are checked by the CHECK_RANGE procedure. The array L(1) to L(10) is loaded with the wavelength of the active diodes. They are spaced with 2 nm on even numbers. On lines 4610 to 4630 the diodes necessary to cover range 1 are activated. The same is done for the range 2 on lines 4640 to 4665. If range 2 overlaps the diodes need not be activated twice (line

4650). The total number is limited to 10. The remaining diodes are used for the normalizing range (lines 4670-4680). If 10 diodes are already exceeded in the second range the procedure is left after having displayed the error message: *'Too large wavelength ranges'* (line 4655).

The other parameters are checked on lines 4700 to 4790 if they exceed certain limits:

Variable	lower limit	upper limit
T(3)	0.0001	10'000
T(4)	0.001	100'000
T(5)	2	40
T(8)	2	10
T(5) * T(8)		200
T(9)	3	1
T(19)	1	5
T(14)	T(4)/100	T(4)/5
T(15)	0.001	1000
T(16)	0.001	1000
T(17)	- 100	100
T(18)	- 100	100
B(1)	1	100
B(2)	1	100
B(7)	.1	10
B(11)	.2	20
B(12)	B(11)	100
B(13)	.2	B(14)
B(14)		T(7) od. 1000

The number of the variables exceeding the limits is passed to the procedure on lines 4795-4799. It displays the array type (T B or W) and the number of the variable for 2 seconds (e.g. variables T3 or 4 are invalid) and sets a flag H2$ = "F". The flag is tested after all variables have been checked. If one or more variables have been found exceeding limits the Main Menu is displayed (GOTO 4000).

AXES_TNS procedure (lines 5000ff)

This procedure determines the X- and the Y-axis of the plot: Turnover number versus substrate concentration. The axes are stored in a file with the file name 'PFTN.KSYS' on the kinetic system diskette. The Y-axis has been entered by the user (A(1,) array). The X-axis starts with 0 as the lowest substrate concentration (X0=0) and ends with the maximal substrate concentration ([S]max) in the assay (X1 = T(14) * 1.05). The initial increment is set to zero (X3=0). The increment X2 is determined from [S]max. so between four and seven digits are written to the X-axis (lines 5030 to 5048). The X-axis is labeled with the name of the substrate (E2$) plus (micro molar) and the Y-axis is labeled with *'Turnover number (1/sec.)'*. The file is generated with the STORE_AXES procedure.

AXES_OD procedure (lines 5100ff)

The axes of this plot are stored in a file with the file name 'PFOD.KSYS' on the kinetic system diskette. The Y-axis is stored as it has been entered by the user. The X-axis is generated using the duration variable T(7). The minimum of the axis is set to zero; the maximum is equal to T(7), the first increment is set to zero. The increment (X2) is set to 2, 5 or 10 seconds depending on the value of T(7) (lines 5110-5130). The X-axis is labeled with *'Time (sec.)'* and the Y-axis is labeled with *'Absorption'*. Again, the file is generated with the STORE_AXES procedure.

AXES_DELOD procedure (lines 5200ff)

The axes of this plot are stored in a file with the file name 'PFAB.KSYS' on the kinetic system diskette. The Y-axis is stored as in the procedure above but with *'Speed (Delta OD/sec.)'* as the label. The X-axis is identical as in the AXES_OD procedure.

STORE_AXES procedure (lines 6000ff)

If the axis file already exists the stored variables are read and compared to the new variables (lines 6005-6010). If there wasn't found a difference, the file needs not to be updated and the procedure stops prematurely. In the other case, the old content is overwritten. First the frame is defined: X-size = 25 cm, Y-size = 16 cm, left rim = 2 cm, and lower rim = 1 cm (line 6030). The variables F5 and F6 are loaded with 1% of X resp. Y dimension in plotter units. The variables P1 and P2 are used to round scalings to approx. 1 %o. Then the file is created, or if it already exists, opened and the axes are stored (line 6105). To this is added the code as it would be sent to the plotter in order to draw the axes (lines 6110-6340). This approach has been selected to relief the measure overlay of the task to generate axes for the plotter. The measure overlay simply read strings from the file and sends them to the plotter.

The plotter is set to default status and pen 1 is selected "IN; SP1;". The size of the plot area is defined with the "IP" command. This area is then scaled by the SC command. On line 6140 the title is plotted. The frame is plotted on lines 6180 and 6190. The loop from line 6200 to 6220 plots the ticks and the digits to the X-axis: The position is converted from user units to plotter units by the equation: I2 = INT((I1-X0)*8000/(X1-X0)+1300), the pen is moved to this position and the tick is drawn "XT,". The digits are plotted with a size of 0.2 * 0.25 cm, the origin being - 1, - 1.2. The other X- and Y-axes are plotted similarly but with different character origins "CP". After the axes have been plotted, the scaling is set to user units using the plotted frame as the maximum plotting area "IP" and "IW" commands. The file is closed and made invisible to the user (line 6360). The routine on line 6400 adds string delimiters to the E5$ string before it is written to disc.

ERROR_RECOVER procedure (lines 9000ff)

This procedure is activated by the 'ON ERROR GOTO 9000' commands if a BASIC command results in an error. The ERROR numbers ERRW are used to inform the user about the type of error that has occured (see lines 9100-9490). After three seconds, the Main Menu is activated. If an unidentified error is encountered, the program stops after having displayed the error number and the error line. The program can be restarted by pressing the 'CONT' key.

4.2 Measure Overlays

Programs that are chained by the start program need to have the identical common data area (see data structures). In addition to that several variables local to the measure overlay are declared on lines 180-210.

4.2.1 Local Variables

F$[1]	= "A"	Flag for automatic or manual operation
F6$[1]	= "0"	Repeat one assay
E3$[30]		X-label of current plot
E4$[30]		Y-label of current plot
F7$[1]	= "0"	Repeat one average "1" or resume "2"
F5$[1]	= "0"	Not linear enough

A$[60]	General purpose buffer
H1$[20]	General scratch buffer for string input
INTEGER I = 0	Loop counter 1
INTEGER I2	Loop counter 2
INTEGER I5	Loop counter 3
INTEGER I6	Loop counter 4
Q1 = 1	1: normal measurement 2: blank measurement
Q2 = 2	1: increasing substrate 2: decreasing substrate
J = 1	Counter repetitions
K = 1	Counter for number of data points
D4$[5,5] = "1" to	Amount of information printed result: 1=all 2=only kinetic Constant
L1 = B(11)	Interval
L2 = B(12)	Integration time
L3 = 256^(1/T(5))	Factor for geometrical row

When the measure overlay receives control it first tests if the common data area has been initialized by the start overlay (C(8) = 1). If not the start overlay is CHAINed (line 800). The first screen of the Measure Menu is shown using lines 500 to 660. As all other menus, it defines a set of 8 softkeys and waits until one of these is pressed

4.2.2 Measure Menu

```
    *** Michaelis const. initial vel.  ***

    Cytochrom c Oxidase    20 nM (E)
```

```
Cytochrome c          250 uM (S)

No. of Points          20
No. of Repetitions      3

Di Var    Text M    [E,S]     Points
Main M    Page 2    Syringe   Wash
```

Softkey #1	displays the header information as it is printed on the printer. To do that, the output to the printer is redirected to the display (PRINTER IS 1) and the PRINT HEADER procedure is called (line 9545). The program pauses so that the screen can be studied by the user. The Measure Menu is shown again, when the 'CONT' key is pressed.
Softkey #2	activates the second screen of the Measure Menu (GOTO 1000).
Softkey #3	activates the the SYRINGE_CONTROL procedure (GOSUB 7250 on line 8609. Upon return from this procedure the Measure Menu is shown again.
Softkey #4	is used to wash the syringes. To do that the WASH_SYR procedure is called.
Softkey #5	returns to the Main Menu. Before the Main Menu can be shown, the start overlay has to be CHAINed (line 800).
Softkey #6	activates the Test Menu. To do that. the 'KONZMSG.KSYS' overlay has to be CHAINed (line 790).
Softkey #7	is used to change the enzyme and the substrate stock concentrations (T(3) /

T(4)). The concentrations, separated by a comma, are input on line 885.

Softkey #8	is used to change the number of data points T(5) and the number of repetitions T(8) (lines 940 and 950).

The second page of the Measure Menu is displayed using lines 1000 to 1260.

```
*** Michaelis const. initial vel. ***
comment :

Syringe 1 : 20 ml   2 : 4 ml   3 : 2 ml
Max. (Substrate)      30 uM
No Blank subtracted
Print on
No reference
Assay Volume          0.4 ml

KmMeas   Blank    [E]auto   Single A
Page 1   Print    Content   Comment
```

If no reference measurement has been measured (C(9) = 0) a beep sounds while the line *'no reference'* is shown.

Softkey #1	jumps directly to the first page of the Measure Menu (GOTO 500). It can be used to exit out of all subsequent routines.
Softkey #2	changes the amount of information that is printed for each automatic determination of a kinetic constant. If the fifth byte of the D4$ string is "1", all information is printed. If this byte is "0" only the file name, date, and the kinetic constants are printed.

Softkey #3	shows the content of the data diskette (line 1600). If a valid file name is entered after the *'Purge file?'* prompt, this data file is erased.
Softkey #4	calls the EDIT_COMMENT procedure to edit the comment (GOSUB 1700 on line 1690).
Softkey #5	activates the routine for the automatical determination of kinetic constants. The sequence on line 1300-1380 is stopped immediately if no reference measurement has been performed (C(9) = 0). The series and a number have to be entered on line 1302. The series D1$ has to be one ASCII character, the number has B1 has to be in the range from 1 to 999. Then the first six characters of the file name are input (D$C15,20J). On line 1310 the number of the syringe to be washed after each automatic determination is copied into the M variable. This syringe is washed if the relative fill level (fill level C(M)) divided by total volumen W(M) is smaller than 12,5%. The number variable of the current determination is loaded D2=B1 and the B1 variable is incremented. If the job queue is empty C(6)=1 the user has to give the enzyme/substrate or inhibitor concentration of the sample that will next be analyzed (procedure BUILD_QUEUE on line 900). Otherwise the header is displayed on the screen (PRINT_HEADER procedure on line 9500). The actual

enzyme concetration (B(3)) is loaded with the preselected enzyme concentration (B(5)). If the enzyme concentration (E(1)) is zero, a blank measurement is performed (Q1=2). In this case the enzyme concentration in the stock is set to 1 and the enzyme concentration in the assay is set to 0.1 (B(3)=0.1). The series character is incremented (A --> B a.s.o.). If the enzyme concentration has been entered with a negative sign, the highest substrate concentration is measured first and the lowest substrate concentration is measured last. This is stored in the variable Q2 while the enzyme concentration is made positive (E(1) = - E(1)). Depending on the syringe that is washed either the enzyme or the substrate stock concentration are loaded from the queue (line 1337). The queue is reduced by one entry and the pointer to the first free element C(6) is decremented by one. The need of enzyme stock per assay is evaluated before the FUBED procedure is called. After the syringes have been filled, the AUTOKIN procedure is called (lines 2000ff). After this routine has been terminated, the status of the job queue is checked and if it is not empty the next kinetic constant is determined using the next sample.

Softkey #6	is used to switch blank subtraction on (D4$ [4,4]="L") and off (D4$L4,4J="0"). If the blank subtraction is switched on, the file name of a valid blank file has to be given. The name consists of the file name (8 characters) plus the extension "N". The file name of the subtracted blank is stored in the header of each data file (D4$ [6,14]).
Softkey #7	selects between automatical dosing of enzyme B(4)=1 or manual dosing of enzyme B(4)=0. In the second case the selected enzyme concentration in the assay (B(5)) has to be given. This concentration is identical to the actual concentration (B(5)). In the automatical dosing mode, the enzyme concentration in the assay (B(5)) is used for the first determination of the acitvity. The actual concentration B(3) is then adapted so that the activity in the assay is between 90% and 110% of the desired activity (B(10)).
Softkey #8	performs a single assay by calling the SINGLE_ASSAY procedure. Before this procedure can be called, the concentration of substrate in the assay has to be given (B(6)). In the same way the concentration of the enzyme in the assay is entered B(5) and B(3) = B(5). The display of the time dependent data is switched off (MODE 0,1) and the single assay is started. If a substrate concentration smaller than zero has been

entered, the SINGLE_ASSAY procedure performs a BASELINE or REFERENCE measurement.

4.2.2 Measure Menu Procedures

BUILD_QUEUE procedure (lines 900ff)

When this procedure is called, it displays the enzyme name B(20)=2 or the substrate / inhibitor name B(20)=3 and the first free element of the queue (C(6)). If B(20) is 2 the prompt looks as follows:

Cytochrome Oxidase Conc. Nr. 5
in Syringe (C=Clear Buffer)
Negative [E] = decreasing [S]?

It then accepts either a number or *"C"* if the queue has to be flushed. The concentration of the enzyme in the tube is stored in the location C(6), in the array E(), (in E(5) in the example above). The pointer C(6) is incremented before the procedure is left.

EDIT_COMMENT procedure (lines 1700ff)

This procedure displays *'Comment 2 lines'* and inputs up to 64 characters (2 lines). If the string is shorter, a space is added (line 1610). The prompt is placed in a way so that the cursor is just on the first character of the old version of the comment so that it need not be typed entirely.

STORE_BLANK procedure (lines 1540ff)

This procedure is called by the End Menu when the D4$[4,4] is set to "L", that means when a blank has been measured. A file with a length of 5 records is created on the system disc and opened. First a part of the header variables is stored (line 1560). Then the delta OD/second information of the two wavelength ranges (arrays T1() and T2()) is stored in this file.

The file is closed and the message *'Blank stored in file ....'* is added to the printout.
SUBTRACT_BLANK procedure (lines 1800ff)
If blank subtraction has been selected, this procedure is called before the first measurement is performed by the data acquisition procedure. It opens the blank file using the name stored in the header of the data file. It then reads through the header variables and compares them with the header variables of the data. If they are not identical, a warning message is displayed by the procedure on line 1885:
Variable; 2; 10 # Blank: 8
Change variable or (999=ignore)
A variable in the file header can be changed or the difference can be ingored. Certain variables must not be changed: If variables 1, 2, 10, 11, 12, 13, or 19 are different the measurement is aborted.

The Blank Correction
A blank has to be measured with the identical parameters as the kinetic constant. Differences between blank and measure parameters cause a warning message. There are a few cases where it is possible to ignore or correct the error condition. In most cases it is advisable to repeat the blank.

List of warnings:
Warning No. 5: 3 blank 1 ?
999<
(example of a warning message with answer

1	: main wavelength range incorrect	-- fatal
2	: main wavelength range incorrect	-- fatal
5	: No. of points are not the same	--please change
8	: No. of repetitions different	-- ignore (999)

9	: dataformat different	-- fatal
10	: internal reference different	-- fatal
11	: internal reference different	-- fatal
12	: wavelength range 2 different	-- fatal
13	: wavelength range 2 different	-- fatal
14	: [S]max different	-- ignore (999)
17	: epsilon different	-- ignore (999)
18	: epsilon different	-- ignore (999)
19	: different row type	-- fatal

If all header variable are correct, the blank is read in and inverted (lines 1850 to 1870). The file is closed and the message *'Blank .... subtracted'* is printed.

MEAS REFERENCE procedure (lines 1900ff)

This procedure flushes the cuvette twice with 0.5 ml of buffer and performs a REFERENCE 10 measurement. The command "*1" is sent by the SEND_CMD procedure (line 7000). It selects syringe 1 as the active syringe. The command "*A0.5" determines the relative amount of buffer to be moved. The command "*B" (GIVE OUT) moves 0.5 ml buffer from the buffer syringe through the mixer into the cuvette. This command is sent a second time after a delay of 2 seconds (line 1905). The reference is measured (line 1920) followed by a test measurement (MEASURE .1). The loops on line 1930 and 1940 wait until the measurements have been completed. The results of the 10 photodiodes are then evaluated. If they are out of range: VALUE(L(J)) > 5), the buffer is added again and the reference measurement is repeated. The reference measurement is important for the diode array detector so that the preamplifiers can be adjusted properly (see HP8451 handbook). If the reference has been completed successfully the flag C(9) is set and the procedure ends.

AUTOKIN procedure (lines 2000ff)
When the AUTOKIN procedure is called first, it resets the status of the photometer (ERASE STATUS). It checks if [S]max (T(14)) is in a valid range. Then the SETUP_KIN procedure is called to test if the file already exists and to multiply the standard substrate concentrations with a factor (L(20)). If a blank has to be used, the subtract blank procedure is called by this procedure. Then the PRINT_HEADER routine is called to print the header information to the printer. The active softkeys (1=Exit 2=Input) are displayed followed by the data headers (DATA_HEADER procedure). At the beginning of the procedure the F7$ flag is set to "0" to indicate serial data acquisition of all different concentrations. Depending on the value of the increase (decrease flag LQ2) either the FOR-NEXT command on line 2245 or 2247 is used.
Next, the buffer for the primary data is cleared (S3()). The variables which are used to add up the nonlinearity are cleared (R9, R8, 56, 57=0) too. T(0) is loaded with the current number of repetitions. This number is higher for the data points with low substrate or inhibitor concentrations because the noise of the average is larger at lower concentrations (see MICHKIN User Manual). On line 2300 the additional number of repetitions (T(0)) is determined. This is added to the usual number of repetitions T(0) = T(0)+T(8) on line 2302 and reduced to 10 if it is too large. The assay is performed T(0) times with the FOR-NEXT loop on lines 2305-2500. In the normal mode, the repeat assay flag (f6$) is set to "0". The assay is mixed and measured by the ASSAY procedure, the derivative is calculated by the DERIVATIVE procedure and the linearity is checked

by the LINCHECK procedure. If the linearity check fails, the assay is repeated (F5$="1" on line 2450). The time dependent data is added up in the S3( ) buffer. R8 and R9 add up the noise information of the single assays. The procedure AVERAGE_VEL is called to calculate the average of 2-10 initial velocities (line 2530). The added up time dependent data is divided by the number of averages. If the debug flag set (F6$="2") the Debug Menu is called before the program proceeds. On line 2595 the plot delta OD versus time is shown on the screen with the PLOT_DATA procedure. The standard deviation among the assays is determined by the CHECK_AVG procedure. If the procedure returns with the FS$ flag set, the Debug Menu is called (GOSUB 4500). If, however, the automatical flag is set (F$="A") the average is repeated. The averages of the two ranges are stored in the result arrays (T(1) T2()) and the results are printed (line 2652). Because overflow during a formatted output generates an error, an unformatted output is selected if one of the variables exceeds 1 or -1 (line 2651). If the repeat single average flag is set, the End Menu is shown. Otherwise the series of assays is finished. Line 2995 shows again the comment (T1$) and the command "#WFCC8" is sent to the ASSAYOMATE. This command is used to read the fill levels of all syringes into the MICHKIN software. On lines 3000-3017 the softkeys of the End Menu are defined and shown on the screen:

4.2.3 End Menu

```
18       20.40        1.0811      0.01587
19  1   -0.002174   -0.003308
19  2   -0.001976   -0.003677
```

```
19  3   -0.001973   -0.003685
19  4   -0.001879   -0.003608
19       24.56       1.0993     0.002779
20  1   -0.002264   -0.003623
20  2   -0.002053   -0.003623
20  3   -0.001965   -0.003425
20  4   -0.002113   -0.003354
20       29.82       1.1231     0.005280

Res M    **END -    MENU*      Syringe
Exit     Output     Rep Avg    Probe R
```

Using the End Menu softkeys, one or several averages can be repeated possibly after a refill of the syringes, a next sample can be added to the job queue, or the last output can be started. If the job queue is not empty (C(6)>1) the next sample is analyzed after the End Menu has been shown for six seconds (line 3020). During this time the user can execute one of the End Menu softkey functions:

Softkey #1	exits directly to the Measure Menu. This softkey is active during the entire data acquisition and processing.
Softkey #2	starts the last output manually if the job queue is empty. In the manual mode F$#"A" or C(6) < 2 the result is stored as a blank file if Q2 =1 by the STORE_BLANK procedure. The blank is thereafter automatically subtracted from all kinetic data unitl manually reset or until the next blank file is measured. The kinetic constants are calculated, the results are printed,

plotted and stored by the OUTKIN procedure.

Softkey #3 repeats one of the averages of several assays with the same substrate concentration. To do that, the repeat single average flag is set (F7$="1", line 3100). The user is asked to select one point. If blank subtraction is on, the inverse value of the blank is read again for the selected point (IF K=J AND F7$="1" THEN T1(K) = -Y0 @ T2(K) = -Y1) by the subroutine on line 3110. The same routine can read all values of the blank starting from the point K if it was called by the resume measure softkey (#5). After the blank has been read the file is closed and the average is measured (GOTO 2255).

Softkey #4 is used to build up the job queue (next probe ready). This is done by the BUILD_QUEUE procedure.

Softkey #5 is needed if a measurement has to be resumed after one or several syringes have been refilled or if a sample has been exchanged. The resume measurement flag is set F7="2" and the point is input (K,P3). The blank is read again by the subroutine on line 3110. Depending on the state of the Q2 flag (1: increasing substrate concentration 2: decreasing substrate concentration) this routine reads either all points larger than K (line 3145) or all points smaller than K (line 3140).

Softkey #6 and #7 perform no function.

Softkey #8 activates the SYRINGE_CONTROL procedure and then jumps back to the End Menu (see below).

If the job queue is not empty, a command is given to the autosampler to change to the next sample. The rest of the current probe is thrown away, and the syringe is filled with the next sample (not in this code). The destination syringe for the new sample is copied into the I variable. If no blank is measured, the fill pointer is adjusted according to the enzyme substrate / inhibitor concentration in the stock (E(1)) and according to the last stock concentration of enzyme T(3) or substrate / inhibitor T(4). Syringe I is activated and the fill pointer is set ("*A"+VAL$(I) and "*="+VAL$(H1)). A wash command with 8 cycles (("*E8", see ASSAYOMATE documentation) is started. If the current sample has been a blank (Q1=2) the blank file is written. The kinetic constant is calculated, the result is printed, plotted and stored by the OUTKIN procedure.

4.2.4 End Menu Procedures

LINCHECK procedure (lines 3300ff)

This procedure calculates the sum of the squares of the differences between the average value (S9) and the delta OD vs. time data (S1(J1)) for the current time window T8 to T9. The root of the mean square is then compared to a minimal value and a relative value. The minimal value (0.00001 * (1+K/T(5)) * B(2)) is approximately the noise of the photometric determination; it increases with the concentration of substrate (K --> T(5)). This minimal noise is made dependent on the value of the linearity variable (B(2))

which is in the range from 1 to 5%. Any assay is accepted that is better than this limit. This ensures that even blank measurements can be accepted by this routine (low noise and low signal to noise ratio). The comparison with the relative value ensures that only the results with a good signal to noise ration or linearity are accepted. If the linearity is not good enough, the message:
'Meas'; K; J; 'Repeated'; Z5 is displayed and the repeat single assay flag (F5$) is set. The Z5 variable counts the failed measurements. If it exceeds three times the number of repetitions, the auto repeat is switched off (F5$="0"), and the results of the single assays are printed (line 3345). It is then up to the user if the measurement can be accepted or if the entire determination of the kinetic constant has to be repeated. If the 'Input' softkey (#2) is pressed during the LINCHECK procedure, the Debug Menu is called before the linearity check is finished (line 3390).
ASSAY procedure (lines 3500ff)
When this routine is called, it calculates the time window (start time and stop time) for the data point (K) which is measured. Two time windows are defined by the user: The time window at the largest substrate concentration and the time window at the smallest substrate concentration. Using these values, the time windows for intermediate concentrations are calculated by linear interpolation:
The times are rounded so that an integer number of primary photometer measurements can be used (measure interval L2). If the first assay of a new concentration (J=1) is selected, the amount of added substrate is calculated and loaded into the ASSAYOMATE by the procedure SFSET (line 7900). For the first data point

of an entire determination (K<2 or K=T(5)) the cuvette is flushed twice before the assay is measured (line 3507). For all assays a stopped flow command is executed ("*H" command), then after a delay which depends on the amount of substrate that has to be added, the time dependence of the absorption is measured (loop from line 3510 to 3570). The average absorption in the normalizing range is calculated by the GETABS procedure (line 4400). If the measurement interval is longer than 0.2 seconds this can be done during the data acquisition, otherwise it has to be done afterwards (lines 3580-3595). Before the new result can be calculated the raw data has to be stored. This is detected by means of the NMEAS flag which contains the number of the measurement in the result buffer. After the end of the data acquisition, the status of the error flag in the status byte of the ASSAYOMATE is checked (I6 = SPOLL (708) @ IF BIT (I6,0) = 1 then 2995). If an error is detected, the data acquisition is stopped and the End Menu is shown. If an overflow error is detected during the data acquisition (ERRN=2), the reference measurement is repeated. In case of another error the program is stopped and the ERROR_RECOVER procedure is activated.
SETUP_KIN procedure (lines 3600ff)
This procedure tests if the file name already exists in the disc directory (lines 3600 and 3605) if it exists, the Measure Menu is shown and the determination of the kinetic constant is stopped. The disc drive is checked if it is switched on and if the proper data diskette is inserted. Then the output devices are selected (PLOTTER1 @ PRINTER 2). The valid spacing type is adjusted so that the highest substrate concentration matches the user defined [S]max (T(14)). To do that,

the procedure SETRE is called (line 8250). A test measurement is executed by the SINGLE_ASSAY procedure (line 4800). The substrate concentration in the test measurement is [Stock]/20. After the single assay procedure, the amounts of buffer, enzyme and substrate/inhibitor are calculated by the FUBED procedure (line 8000). The print header procedure is called with the parameter H1. If it is set to 1 the entire header is printed otherwise only the file name and date are printed. The data arrays (T1() T2()) are cleared and the SUBTRACT_BLANK procedure is called if necessary.

DATA_HEADER procedure (lines 3700ff)

This procedure prints and displays the column headers according to the value of the printer flag. It calculates and displays the approx. duration of the measurement. It clears the *'repeat one assay'* flag (F6$).

AVERAGE_VEL procedure (lines 3750ff)

This procedure calculates the average velocity of several assays (T1(41) = S4(), T2(41) = S5()). It displays the data point number (K), the substrate concentration (H4), the actual enzyme concentration in the assay B(3), and the average nonlinearity of the assays.

CHECK_AVG procedure (lines 3800ff)

This procedure adds up the squares of the differences of the data from the mean value. The root of the sum is then compared with the noise variable (B(1)). The data is accepted when the absolute noise is very small (0.00005 * B(1)) or when the relative noise T1(41) / 100 * B(1) is smaller than the value specified by the B(1) variable. If the relative and the absolute noise are too large, all the assays for the average are repeated and the *'failed measurements counter'* (Z5) is incremented. If this variable is larger than 2 * T(8) the average is not repeated but the message *'Average nonreliable'* is added and the row data is printed (lines 3870 and 3875).

OUTKIN procedure (lines 4200ff)

If the printer flag (D4$[5,5]) is set, the Delta OD vs. time plot is labelled (E4$) and the graphic screen is copied to the printer. The PRINT_HEADER procedure is called once more to repeat file name and date of measurement (line 4202). Next the columns for the turnover number versus substrate concentration table are labelled. The labels and axes for the turnover number versus substrate concentration plot are read from the file and shown on the screen (lines 4230 and 4250). The data file is opened by the OPEN_DFILE procedure. The loop from lines 4270 to 4320 plots, prints, and stores the results of the two ranges. The procedure CALCTN is called to return the substrate (H4) and enzyme (H2) concentrations, and to convert the velocity data into turnover numbers (H3 and J1). On line 4290 both ranges are plotted. On line 4310 the result is printed in the following format:

```
   21    +75.1          +74.8          24.28
```

On line 4312 the result is wirtten to disc. If the printer flag is set the plot is copied to the printout. Then the file is closed and the routine LINRE@ is called to calculate the kinetic constant.

GETABS procedure (lines 4400ff)

This routine averages the absorption of several photodiodes in range 1 and 2. The average of the photodiodes from the normalizing range is subtracted from the results in S1() and S2().

DEBUG_MENU procedure (lines 4450ff)

This routine enables the user to influence the data acquisition. This menu is called after the softkey #2 'INPUT' has been pressed. It offers 8 functions:

1	Exit	Resumes data acquisition with no action taken
2	New Stdev.	Prompts for new values for the maximal nonlinerarity resp. maximal noise (B(1), B(2) line 4560).
3	Rep. Single	The number of the single measurement has to be selected (line 4530). The result buffer S3() is cleared and the rep. single flag is set.
4	Rep. Aver.	This function starts the substrate concentration currently being measured with the first assay again.
5	End Me	Jump to the End Menu without measuring the data points.
6	Go on	Resumes data acquisition with no action taken.
7	Test Msg	Jumps back to the single assay procedure for a new test measurement.
8	Next ready	This function is used to add another sample to the job queue. To do that the procedure BUILD_QUEUE is called.

DERIVATIVE procedure (lines 4600ff)

This procedure derivates the absorption versus time data so that the S1() and S2() arrays contain delta OD versus time information (delta OD/per second). The average velocities are added up for the current time window (S9 S8 line 4700) and divided by the number of data points. The velocities for the two ranges are shown on the display as follows:

```
    17           2          -0.000318      -0.00175
average        assayno.    velocity 1     veolcity 2
```

The velocity are then copied into the arrays S4() and S5().

4.2.5 SINGLE_ASSAY Menu

SINGLE_ASSAY procedure (lines 4800ff)

10 photodiodes are selected on line 4810. The timescale of the plot is set from 0 to the stop time at the highest substrate concentration T(7). The enzyme and substrate concentrations in the assay are checked and set to stock/3 if they are too large. The volumes are calculated on lines 4835 and 4836. A negative substrate concentration triggers a reference measurement (line 4840). If no reference measurement has been performed C(9) = 0 the routine is stopped and the Measure Menu is displayed. The desired assay is loaded into the ASSAYOMATE by means of the SETSF procedure and the assay is executed with the "*H" command (line 4860). The ASSAY procedure is called for an intermediate data point and the time window is set to B(13) resp. T(7). The average absorption of the first four data points is stored in the H1 and H2 variables. The axes of the OD versus time plot are loaded. The optical density is used to calculate the substrate concentration which is shown on line 5057. The derivative procedure is called to calculate the derivative of the absorption which is then checked by the LINCHECK procedure. The relative noise is calculated from the absolute noise 56 and the average velocity S9. The absolute and the relative noise are shown on line 5160. The linearity is

calculated by comparing the velocity of the first five data points (A1 and H4) with the velocity of the last five data points H2. The relative nonlinearity (H3) is then displayed on line 5190. If the print flag is set the Single Assay Menu is shown:

```
*******   Single Assay Menu    *******
Syringe   1    is    filled
K* .34226,.037736,.02
Initial OD  Range 1 and   Range
             +0.2077       +0.41164
[S]          +2.308        +2.713
Deriv.      Range 1 and   Range 2
 0,  0       +0.00541     +0.00148
Noise Range 1: 0.000069   =12.7%
Nonlinearity :            =10%

[E]       Activ.    Syringe    [S]
Exit      Go on     Graph      Alpha
```

Softkey #1	is used to exit to the Measure Menu. It is active during the entire AUTOKIN and SINGLE_ASSAY procedure.
Softkey #2	switches back to automatical mode after the user has changed anything manually in the Single Assay Menu.
Softkey #3	is used to show the graphic screen of the menu and to wait in manual mode for the next user command (line 5420).
Softkey #4	is used to show the alphanumeric screen of the menu and then waits in manual mode for the next user command (line 543)).

Softkey #5	is used to change the enzyme concentration in the assay. It shows the active enzyme concentration B(3) and prompts *'new?'*. This enzyme concentration is then used as the preselected enzyme concentration (B(5)). The variable J1 is set to 9; that causes the single assay to be repeated (lines 5310 and 5390).
Softkey #6	is used to change the desired velocity (B(10)) which is used to determine the enzyme concentration in the assay in the automatic enzyme mode.
Softkey #7	is used to change the content of one syringe by calling the syringe control procedure (Gosub 7250 on line 5495).

The amount of enzyme is adjusted automatically when this mode is selected (B(4)=1) and when no blank is measured (Q1=1). If the velocity, or activity, is less than 50% of the desired velocity, the enzyme concentration is doubled (line 5455). If it is larger than twice the desired velocity it is reduced to half of its former value (line 5460). If the difference is more than approximately 30% it is corrected by this amount (lines 5465 and 5470). In all these four cases the repeat assay flag is set (J1=9).

Upon returning from the menu subroutine, the repeat assay flag is checked and the assay is repeated if necessary. Then the delta OD versus time plot is read from diskette and shown on the screen. The data is added by means of the PLOT_DATA procedure and the menu subroutine is called once again (line 5310-5390).

4.2.5 Store, Plot, and Linreg Procedures

OPEN_DFILE procedure (lines 5500ff)

This procedure creates a file on the data diskette (B9$) with the file name D4$[15,20] and the number D2. The file is opened and the header information is written (line 5570), see data structures).

LINREG procedure (lines 6000ff)

This procedure performs a linear regression to the double reciprocal transformed data (H1=1 / substrate concentration, H2=1/turnover number).

A1	= loop counter N
A2	= Sum X
A3	= Sum Y
A4	= Sum X*Y
A5	= Sum X^2
A6	= Sum Y^2

The parameters of the linear regression are then calculated:

H1	rho X	= Sum X^2/N - (Sum X/N)^2
H3	intercept	= (Sum X*Y/N - Sum X/N * Sum Y/N) / H1
H4	slope	= Sum Y/N - H3 * Sum X/N
correlation		= (slope * (rho X)$^{-2}$) / (rho Y)$^{-2}$

The kinetic constants velocity and Michaelis constant are calculated and printed on lines 6280-6310. The correlation is added on line 6320.

PLOT_DATA procedure (lines 6500ff)

This procedure plots the contents of the arrays S1(), S2(), or S3() depending on the flag F6. The PLOT H4, H3, 1 command plots a line from the old position to the current position. The plot pen is lefted before the routine is left.

4.2.6 Procedures for ASSAYOMATE Control

SEND_CMD procedure (lines 7000ff)

Before a command is sent to the ASSAYOMATE, an IEEE timeout is set to 3 seconds (lines 7005 and 7010). The status of the ASSAYOMATE is read with the I5=SPOLL

(708) command. The status byte is analyzed on lines 7025 to 7035: Bit 7 is the busy byte, a next serial poll is performed after a delay of 200 milliseconds. A comma is added to the command string A$ and the string is sent with the OUTPUT 708: A$ command. Lines 7052 to 7065 are used to keep track of the fill levels of the syringes with no need to read them from the ASSAYOMATE after each command. If the empty flag is set no give out or empty commands are sent to the ASSAYOMATE (line 7082). In case of an IEEE timeout the message *'Switch on ASSAYOMATE'* is displayed and the user is warned with a sound. The IEEE interface is reset and a next serial poll is attempted.

SETSF procedure (lines 7100ff).

The variable W1(4) contains the number of primary cycles or the speed of the stopped flow routine this number is converted into hexadecimal and added to the LDHEME command (*TF84A, load ASSAYOMATE memory with hex information). The amounts of added buffer, enzyme and substrate are converted into waiting cycles (lines 7110-7125) and added to the command string in hexadecimal form. This string is sent to the ASSAYOMATE by means of the SEND_CMD procedure. The procedure on lines 7200-7220 converts integer numbers into four byte hexadecimal numbers.

SYRINGE_CONTROL procedure (lines 7250ff)

The *WFCC8 command triggers the ASSAYOMATE to send the menu line with the fill levels to the computer. The ASCII information is received and interpreted by the RECEIVE_FILL procedure. Thus, the accurate fill levels of all syringes are known by the MICHKIN software each time the Syringe Control Menu is called. The menu does not show a screen of its own only a set of 8 softkeys which performs functions implemented in the ASSAYOMATE

Syringe Control Menu. The Syringe Control Menu is just the user interface of these functions in the MICHKIN software.

Softkey	command	label	parameter
1	*B	GIVE OUT	amount in field 5
2	*C	SUCK IN	amount in field 5
3	*D	GIVE BACK	amount in field 5
4	*F	EMPTY	fill level 0
5	*A amount	0.5	sets the amount per key variable
6	*G	FILL LEVEL	the softkey shows the fill level. When it is pressed the syringe is filled
7		RETUR	exits
8	*1 *2 *3	SYR.1	selects the next syringe if the softkey is pressed (lines 7480 and 7485).

WASH procedure (lines 7500ff)
This procedure is called from the Measure Menu. It prompts for the number of cycles and for the syringe (1-3). On line 7520 the select syringe command is sent and on line 7530 the wash syringe command is sent. The fill level after washing is zero (odd number of cycles) or the maximal fill level (W(M+5), even number of cycles).
RECEIVE_FILL procedure (lines 7700ff)
As for the sending procedure a timeout is defined to avoid bus hangups. Binary data is input by the ENTER 708 "#;B" : H1 statement. The variable H1 is converted into ASCII and added up to the string A$ until a

carriage return is met or the length of the string exceeds 40 bytes. The timeout is switched off and the sub strings 10-16, 20-26, and 30-36 are converted into real numbers C(1), C(2), and C(3). In case of an error, the ERROR_RECOVER procedure is acitvated.

SFSET procedure (lines 7900ff)

This procedure provides the volumes of buffer, enzyme, and substrate/inhibitor when it is called with the parameter K (number of data point). Before this routine can be used, the spacing type T(19), the factor L(20) and the assay volume B(7) have to be defined. The volumes are calculated such, that they can be titrated by the STOPFL (*H) command.

Spacing type	line	type
1	7920	geometrical
2	7930	arithmetical
3	7950	reciprocal
4	7940	mixed
5	7960	constant

Examples of the substrate concentration provided by the spacing types are given in the MICHKIN user manual. The routines on lines 7920-7960 provide a fixed spacing which has to be mulitplied by a factor: W1(3) - N3* L(20)/10000. The enzyme concentration can be varied in a linear fashion (arithmetic) by defining a slope B(10). The enzyme concentrations are reduced to 50% if B(10) is set to 0.5. The amount of buffer calculated on line 7980. If the SFSET procedure was called with I set to 9 it is stopped on this line. Otherwise the assay mixture is sent to the ASSAYOMATE by means of the *K command.

FUBED procedure (lines 8000ff)

The current fill level of all syringes is read by the *WFCC8 command and the RECEIVE_FILL procedure. The amounts of buffer enzyme and substrate needed for the acquisition of all data points is calculated by the loop on line 8010. A minimal fill level of 4 ml, 0.3 ml, 0.25 ml is given for the buffer, enzyme, resp. . substrate syringe. For each data point an additonal margin of 10% is added to allow a limited amount of repeats with no need to refill during the data acquisition. If the needed amount exceeds the current fill level S1(M) > C(M) the syringe M is filled by the subroutine on line 8100. This subroutine activates syringe M and fills the syringe by means of the *G command. The syringe which is automatically refilled from the sample changer B(20) is filled to the desired amount if the available amount in the sample tube (B(19)) is large enough. The fill level is set by the '*=2.7;' command before the fill command *G is sent.

CALCTN procedure (lines 8200ff)

The procedure SFSET is called either with the parameter I set to 9 (no. sending to ASSAYOMATE) or to ? (sending to ASSAYOMATE). Using the columns of buffer, enzyme, and substrate provided by substrate concentration (H4) are calculated. If T1(K) and T2(K) contain valid velocity results, the turnover numbers of range 1 (H3) and of range 2 (J1) are calculated on line 8230.

DIS_AXES procedure (lines 8500ff)

This procedure reads the X- and Y-axis together with the label E3$ and E4$ from the file A$. The frame and the ticks are displayed by the XAXIS and YAXIS commands (see HP85 manual). On lines 8660-8680 the digits are plotted below the ticks. Similarly the digits are plotted to the Y-axis on lines 8690 to 8710. The file is closed and the procedure ends on line 8900.

PRINT_HEADER procedure (lines 9500ff)

The output generated by this routine depends on the state of the flag H1. If H1 is zero the output is short:

```
      Measurement  :    HORSCC  B  5
      6. Nov. 86  Volume    :   Modcyt
      Michaelis constant of Oxidase  VIA
    in 0.2%  Tween  80  at  150 mM  I.str.
```

If H1 is 1 a longer output is printed:

```
      Measurement   :    HORSCC  B  5
      6. Nov. 86  Volume    :    Modcyt
      Michaelis constant determination
      Time window:
      at [S]min          .4   to   1.6 sec
      at [S]max         3.2   to   8   sec
      Interval           .4   Integration  .4
      Maximal Noise                      5%
      lambda-range 1 from     549 to 552 nm
      lambda-range 2 from     418 to 420 nm
      Internal ref. from      580 to 586 nm
      [cytochrome oxidase]         : 30  nm
      conc. in assay               :  3  nm
      [cytochrome c]          : 250 micro-mMax.
      conc. in assay     :  30 micro-m
      No. of points      20        Rep.   3
      Delta epsilon      1    : -16.2 l/(mM cm)
      Delta epsilon      2    : -43.1 l/(mM cm)
          Assay volume                      .4 ml
```

4.3 Testmeasure Overlay 'KONZMSG'

The 'KONZMSG' overlay has the same common data area as the other overlays. The following variables local to this overlay are defined.

4.3.1 Local Variables

```
A$[60]      I/O buffer
P9$[20] = 'Result:'                 text for initial
                                    velocity
P8$[20] = '1st order constant:      'text for 1st order
                                    constant
H1$[64]     general purpose buffer
F$[1]       E: single assay
            M: serial assay
F1$[1]
V[15]       array for assay variables for Testmeasure
            Menu and Serial Assay Menu
I,I5,I6     loop counters INTEGER
W1=400      wavelength range for spectrum
W2=800      wavelength range for spectrum
T6=0        start time
T7=20       stop time
T8=5
H5=20       amount of substrate in ml
H6=10       amount of enzyme in ul
L1=5        measure interval
L2=5        integration time
V(1)=1      1=Absorption 2=Derivative
V(2)=0      printout 1=initial velocity 2=1st order
            constant
V(3)=550    measure wavelength
V(4)=0      minimum of Y-axis
V(5)=1      maximum of Y-axis
V(6)=0
V(7)=10     maximal length of measurement
V(8)=2      start of time window for target
V(9)=5      ent of time window for target
V(10)=0     initial veloctiy
```

V(11)=1	conversion factor Delta OD/sec./turnover number
V(12)=0	first order constant
V(13)=1	conversion factor
V(14)=0	-
V(15)=0	-
S3(70)=1	counter of measurements
S3(71)=0	storage file number
S3(72)=0	transmit flag
S3(74)=2	syringe with variable probe
S3(75)=0	number of different assays

After these variables have been initialized, the *'common area initialize flag'* C(8) is tested and the 'Autost' overlay is CHAINed (lines 320, 800, and 805) if it is different from 1. The actual fill level flag C(15) is tested to decide if it is necessary to read the fill levels of the syringes from the ASSAYOMATE. On line 492 the maximal fill levels of the MICHKIN software are loaded down into the ASSAYOMATE.

4.3.2 Test Menu (lines 500ff)

When this menu is active the following screen is shown:

```
***********     Test Menu     *********
***********   2nd level menu  *********
Please select a softkey :
Test M  = Test measurement
Auto M  = Measure sampleas repeatedly
Syringe = Manual moving of syringes
Wash    = Washing of syringes
Exit    = Emergency exit
Refer.  = Reference measurement
(before first measurement)
```

Main M	Refer.	Auto M	Test M
Exit	Conc. M	Syringe	Wash

Softkey #1	is used as an emergency exit in all subsequent menus and procedures. When it is pressed , the Test Menu is shown and the function is aborted.
Softkey #2	activates the Concentration Menu (lines 1000ff)
Softkey #3	reads the current fill level of the syringes from the ASSAYOMATE "*WFCC8" @GOSUB 7000 @ GOSUB 7700 and shows the Syringe Control Menu (syringe control procedure is called).
Softkey #4	is used to wash the syringes. To do that the WASH procedure is called.
Softkey #5	chains the Start Overlay (lines 800 and 805).
Softkey #6	executes a reference measurement. To do that MEAS_REFERENCE procedure is called.
Softkey #7	acitvates the Serial Assay Menu (lines 9000ff).
Softkey #8	activates the Test Menu (lines 4000ff).

The message *'Basline not yet measured'* is only added if a baseline has to be measured (see HP 8451 Manual) why baseline or reference measurements are necessary. Any measurement still in progress is stopped with the STOP MEASURE command before the menu is displayed. (lines 4000ff).

4.3.3 Measure Concentration Menu
This menu provides functions to mix assays and to measure a spectrum in order to determine the

concentration of a substrate or of a product. This menu appears to the user as follows:

```
****  Measure Concentration Menu  *****
Range 1     from    548   to   552 nm
Range 2     from    416   to   420 nm
   Int. Ref.   from    580   to   586 nm
Epsilon value Range 1 and Range 2
Substrate :     25.2   and  121.1   1/nM cm
Product   :      9.0   and   78.0   1/nM cm
Assay volume     0.4 ml
Spectrum    from    400   to   650 nm

Fill level     Syr. 1    Syr. 2    Syr. 3
               35.00      1.25      0.865

Assay V     Spectr.      [E,S]     L-Range
Exit        Substr. M    Prod M    Epsilon
```

Softkey #1	exits to the Test Menu.
Softkey #2	starts the concentration determination of the substrate. This is done by the MEASURE_SUBSTRATE sequence from line 2000 to line 2490.
Softkey #3	starts the concentration determination of the product. This is done by the MEASURE_PRODUCT sequence from line 2500 to line 2890. This sequence is similar to the measure substrate sequence but enzyme is added to the assay and product extinction coefficients are used.
Softkey #4	is used to change the extinction coefficients. The following message is shown before the coefficients have to be

typed by the user:

Please enter the extinction coefficients for wavelength range 1 :
First the substrate, then the product, the two entries have to be separated by a comma
29.7,8<
Please enter the extinction coefficients for wavelength range 2 :
First the substrate, then the product, the two entries have to be separated by a comma
120,88.1<

First the exticition coefficients for range 1 have to be given (T(15) and T(16) on line 1355), then the extinction coefficients for range 2 have to be given (H1 and H2 on line 1365). The coefficients are converted into coefficient of substrate and difference substrate minus product T(17) = T(15)-H1 and T(18) = T(16)-H2.

Softkey #5 is used to change the assay volume. The assay volume B(7) is accepted if it is between 0.2 and 1.5 ml (line 1310). Otherwise, the prompt is repeated and the input has to be given again.

Softkey #6 starts the measurement of a spectrum (line 1500). The photometer is set to the proper conditions:

ERASE STATUS	activates default conditions
LAMBDA W1 to W2	wavelength range
Y-SCALE	automatic Y-scaling.

	Before the spectrum is measured a new mixture with the composition shown in the menu can be performed. In this case a reference is measured (line 1900), the *K command is used to load the new mixture (*K H5/1000, H6/1000, B(7)-W1(3)). The *H command is sent to the execute a stopped flow addition. This is repeated and a spectrum is measured after a delay of four seconds. Then the PHOTOMETER_MENU on line 5400 is activated. This menu is used to switch between alphanumeric and graphic screen and to type manual commands to the photometer.
Softkey #7	inputs the new amounts of enzyme and substrate in ul (H5 and H6). The volumes are tested if they are smaller than 1000 ul (line 1410).
Softkey #8	is used to change the wavelength range for the spectrum (W1 and W2). The entries are tested on line 1422.

ERROR_RECOVER procedure (lines 1660ff)
This procedure shows the error number and the error line and waits that the user corrects the error condition. A CONT <line number> command restarts the program.

REFERENCE procedure (lines 1900ff)
This procedure has been described in the KIN2 overlay.
MEASURE SUBSTRATE procedure (lines 2000ff)
The desired mixture is loaded down into the ASSAYOMATE (line 2050). Within the loop from line 2120 to line 2220 a stopped flow command is sent and the STABILIZE

procedure is called to wait until the absorption has become stable. When the user has pressed the RESULT key in this procedure, the absorptions at range 1 and 2 are returned in the H1 and H2 variables. The result is printed and the absorptions are stored in the arrays S4( ) and S5( ). This is repeated T(8) times before the average absorption is calculated (line 2660) and printed. The concentration of the substrate is claculated on line 2320 and printed on the next line. After the measurements the printout looks as follows:

```
Concentration measured at the
substrateabsorption
dilutionfactor  1 : 20

No.     Range 1     Range 2
 1      +.03737     +.29215
 2      +.04045     +.30737
 3      +.04020     +.29809
average of 3 Measurements :
        +.03950     +.29876
Cytochrome c concentration
in the substrate syringe :
        80.39 µM    79.90 µM
```

MEASURE_PRODUCT procedure (lines 2500ff)
This procedure is closely similar to the procedure above. Differences are found on line 2560:
W1(2) = H6/1000 /enzyme concentration is not zero) and on line 2740: Extinction coefficient of the product is T(15) + T(17) and T(16 + T(18).
STABILIZE procedure (lines 3000ff)
This measures a spectrum from W1 to W2 without displaying the result (MODE 0.1). The duration T(5)

together with the flag Q1=0 is passed to this procedures by the calling procedures. The average absorption of range 1 and range 2 is calculated on lines 3120 to 3190 after the NMEAS flag has been set. The absorption is shown on the screen on lines 3201 to 3220. The absorption is measured and displayed until the flag Q1 is set by the user (softkey #2). In this case a five second measurement (T5=5) is performed before the procedure ends. If softkey #3 is pressed the spectrum is shown on the display (MODE 0.0) on lines 3260 to 3270.

4.3.4 Testmeasure Menu (lines 4000ff)

Two sets of softkeys are provided in this menu to measure assays and to determine initial velocity and first order constants. The menu screen appears as follows:

```
************* Test Measure  ***********
Function     : ABSORPTION
Y-Scale      : 0   to   1   OD
Measure      : 0   to   10   sec
Tangent      : 2   to   5   sec
Mixture      : Buffer 320 ul
Enzyme 40 ul  Substrate 40 ul
Delta OD/sec: -0.0145      Fact. -100
Result       : 1.45
1st order constant:
Fill level     Syr. 1    Syr. 2    Syr. 3
             35.00      1.25      0.865

Pl axis     Tangent     Factor     Store
Print       Time        Graph      Page 2
```

1st set of softkeys:

Softkey #1	exits to the Test Menu (GOTO 500).
Softkey #2	is used to define a new mixture (user input in ul is converted into ml). Buffer 0-2 W1(1) Enzyme 0-1 W1(2) Substrate 0-0.5 W1(3)
Softkey #3	executes on assay. To do this, the TEST_ASSAY procedure (line 5000) is called. This function can be executed after all variables in the menu have been set to proper values.
Softkey #4	selects the second softkey set (GOTO 4300).
Softkey #5	plots the last result on the plotter. To do that, the linetype (H1) has to be selected. The IEEE plotter with the address 5 is selected on line 4457, the linetype is selected line 4460 and the line is plotted (line 4462). The display is selected as destination device before the function ends. If the plotter is not switched on, an error message is shown (line 4470) and the command is ignored.
Softkey #6	is used to toggle between absorption V(1) = 1 and derivative V(1) = 2 function.
Softkey #7	is used to select the wavelength for the measurement (V(3) on lines 4550-4560). The input is tested if it is an even number.

Softkey #8	defines the Y-scale of of the display with two numbers V(4) and V(5). If both numbers have the same value (e.g. 0.0), the Y-axis is scaled automatically and only one data curve is shown on the screen. In all other cases the axes are drawn by the OVERLAY 0, V(7), V(4), V(5) command and al subsequent data is drawn on the same plot.

2nd set of softkeys:

Softkey #1	selects the type of printout V(2). If a printout is selected (V(2)=1 or 2) the appropriate message P9$ or P8$ has to be typed in.
Softkey #2	is used to enter the maximal duration of the measurement V(7). Depending on the value different intergration time and time interval are selected. If fixed scaling of the Y-axis is selected V(4) # V(5), the modified axes are displayed (line 4675).
Softkey #3	pressing this softkey shows the freshly updated graphic screen (lines 5250 - 5390). Depending on V(1) either ABSORBANCE or DERIVATIVE is selected. If V(5) = V(4) the commands Y-scale and TIME SCALE 0 TO V(7) select fixed X- and automatic Y-scaling. The command PLOTTER on line 5310 displays the data. The kinetic constant is calculated by the KINCONST procedure (line 4900) before the softkey function ends.
Softkey #4	selects the first softkey set (GOTO 4200)

Softkey #5	plots the axes on the IEEE plotter (705) (lines 4480-4495).
Softkey #6	is used to change the start (V(8)) and stop (V(9)) time for the analysis time window (tangent). The stop time has to be shorter than the maximal duration of the measurement (line 4710). Again, the KINCONST procedure is called to recalculate the kinetic constant.
Softkey #7	is used to change the conversion factor V(11) for the conversion of delta OD/sec. into turnover nuber or units /sec. (line 4750).
Softkey #8	selects storage and transmit options (lines 6000-6045). The absorption versus time data is stored in a file on the data diskette if a number in the range from 1 to 999 is entered after the prompt on line 6010. The absorption versus time data is sent to the host if the send flag S3(72) is set. On line 6042 a comment which will be sent together with the data can be entered. The first 8 characters of this comment are stored on disk if storage is selected. The SSOPTION procedure on line 6050 calls the STORE_KIN or the SEND_KIN procedure to store resp. send the data.

KINCONST procedure (lines 4900ff)

The loop from line 4910 to 4925 adds up the velocity data over the time window. If the derivative has already been performed on the source data V(1) = 2 it needn't be done in this loop. On line 4930 the rounded average initial velocity is stored in the variable

V(10). On lines 4940 to 4980 the 1st order rate constant is determined with the Guggenheim method: This can only be done if the ABSORPTION function is selected.
First, the natural logarithm of the absorption difference (OD(T) - OD(T+deltaT)) is calculated (LOG(H3) and LOG(H4)). The average slope first order constant of the logarithmic plot is determined by averaging (H4-H3)/L1. The first order constant is stored in the variable V(12) (line 4980). An error leads to the abortion of the calculation (line 4997).
TEST_ASSAY procedure (line 5000ff)
After this procedure has been called, the user has to enter the number of repetitions (R). The entry is accepted if it is in the range from 1 to 10. The message *'*** The assay is mixed *******'* is displayed and the SERIAL_ASSAY procedure is called to execute the assay.
SERIAL_ASSAY procedure (lines 5010ff)
This procedure selects the display as the output device. In the autoscaling mode, the Y-axis is set from 0.01 to 1.5 before the result is known line 5030. The measure wavelength is selected and the assay mixture is loaded into the ASSAYOMATE. A first assay is mixed but not analyzed (line 5045). If a printout is selected the mixture is printed. The loop from 5065 to 5090 repeats the assay R times. On lines 5070 and 5075 the assay mixcommand is sent and the measurement is started after a delay of 1.5 seconds. The delay is increased by one second if fixed scaling is active (line 5072). The data is shown on the display (MODE 0.0) and stored in HP 8451 memory after the last data point has been measured (NMEAS = V(7)/L1 +1). The status byte of the ASSAYOMATE is read. If the next probe ready bit (BIT(X,3)) is set

and the procedure was called by the Serial Assay Menu F$="M" and the last repeat has been measured, the next probe is filled in syringe (S3(74), line 5085). The kinetic constant is calculated with the KINCONST procedure. Lines 5092 to 5130 are needed to average R assay (HP 8451 syntax). The result is derivative if DERIVATIVE function V(1) = 2 is selected and the plot is redrawn in the autoscaling mode (lines 5150-5170). The kinetic constant for the averaged data is calculated and printed:

```
Mix:   B   320   E   40   S   40   ul
Average:
Turnover number              157.8643
```

PHOTOMETER Menu (lines 5400ff)
This small menu provides a set of softkeys to show the graphic screen (line 5410) to go on (lines 5490) and to exit to the BASIC interpreter (line 5500).
SSOPTION procedure (lines 6050ff)
If the S3(71) variable is not zero, the STORE_KIN procedure is called. If S3(72) is one the SEND_KIN procedure is called.
STORE_KIN procedure (lines 6100ff)
The data is stored on diskette with functions described in the HP 8451 manual (TO FILE H1).
SEND_KIN procedure (lines 6200ff)
On lines 6210 to 6270 the RS 232 interface is set to proper conditions by means of the CONTROL 10, ... statement (I/O ROM of HP 85). On lines 6280 and 6290 a timeout prevents bus hangups. If a timeout is encountered, the current transfer is aborted (line 6400). First the comment is sent after the RS 232 device has been selected

CONTROL 10,2 ; 7	Modem lines are set
CONTROL 10,3 ; 15	Baud rate 9600
CONTROL 10,4 ; 7	8 bits 2 stop bits no parity
CONTROL 10,5 ; 48	Select handshake
CONTROL 10,11 ;192	XON/XOFF handshake
CONTROL 10,14 ; 19	XON/XOFF handshake
CONTROL 10,15 ; 17	XON/XOFF handshake

as a printer. The same printer is selected as the output device for photometer data (PRINTER 15 10). The data is sent to the host by executing a PRINTER command (line 6320). The timeout is cleared and the internal printer is selected again before the procedure ends.

SEND_CMD procedure	lines 7000 ff
SETSF procedure	lines 7100 ff
SYRINGE_CONTROL procedure	lines 7250 ff
WASH procedure	lines 7500 ff
RECEIVE_FILL procedure	lines 7700 ff
SFSET procedure	lines 7900 ff
FUBED procedure	lines 8000 ff

(See measure overlay for description of these procedures)

4.3.5 SERIAL ASSAY Menu

The softkeys in this menu provide functions for the automatical determination of several assays on series of samples which are changed by the autosampler. The screen appears to the user as follows:

```
************* Auto Measure  ***********
Function     : ABSORPTION
Title of Job:
   Number of Measurement :  1
Storage                 :  off
```

```
Sending              :  off
Plotting             :  off
Wash syringe         :  2
Fill level           :  5 ml

Fill level    Syr. 1   Syr. 2   Syr. 3
             35.00    1.25     0.865

Title      Number     Store      Plot
Exit       Job        Meas       Wash
```

Softkey #1	exits to the Test Menu.
Softkey #2	is used to build a job composed of one or several assays with different composition with the same sample. On line 9410 the input buffer S3( ) is cleared. The input loop from line 9450 to 9465 accepts n times four entries: Number of repetition 53(I1), amount of buffer S3(I1+1), amount of enzyme S3(I1+2), and amount of substrate S3(I1+3). The prompt is repeated until 0 repetitions are entered or I1 is larger than 51. The variable S3(75) is loaded with the numer of different assays (I).
Softkey #3	executes the job that has been entered using softkey #2. The F$ flag is set to "M" to select serial assay mode in the serial assay procedure. F1$ is set to "N". The number of the measurement, the date and the comment are printed. The counter 53(73) is incremented and V(2)=1 switches printing on. The message

'Probe 1 sucked in
Press key #9 at the
ASSAYOMATE, when the
next probe is ready'

is displayed and the next probe ready bit in the ASSAYOMATE status byte is tested (line 9525). If no probe is ready the routine stops and the Serial Assay Menu is shown. If the bit has been set, the selected syringe is filled wiht the new sample (line 9530).
Then the message

'Prepare probe 2
within the next 2 minutes'

is shown (line 9540). The loop from line 9550 to line 9585 executes the assay in the batch job one after the other. S3(75) contains the number of assays that are stored in the batch buffer. The variables R, W1(1), W1(2), and W1(3) are loaded from the batch buffer and the serial assay procedure is executed. After the measurement the result is stored or sent with the SSOPTION procedure. If plott in is selected (S3(73)=1) the PLOT_KIN procedure is called to plot the OD vs. time data. The linetype is incremented after each measurement but reset to 1 if it exceeds 8. The loop is executed until the next probe bit is no longer set.

Softkey #4 is used to select which syringe is to accept the different probes (S3(74), line 9610).

Softkey #5	is used to enter the title T1$ of the job (line 9650).
Softkey #6	is used to set the number of the measurement (S*(70), line 9700).
Softkey #7	is used to change the storage and sending of data.
Softkey #8	is used to switch plotting on and off (S3(73), line 9800).

4.4 Data transfer Overlay 'DACOM'

The 'DACOM' overlay has the same common data area as the other overlays. The following variables local to this overlay are defined:

4.4.1 Local Variables

2$[2000]	IOBUFFER
B8$[64] = " "	File name on host
H1$[64]	General purpose buffer
F$[1]	Flag for automatical manual operation
F6$[1] = "0"	Repeat one assay
E3$[30]	X label
E4$[30]	Y label
F7$[1] = "0"	Repeat one average
ES$[60] = " ....."	Title
N9$[30]	Names of blank files for average
Y0=0, Y1=1, Y2=1, Y3=0	Y-axis for plot
X0=0, X1=1, X2=1, X3=0	X-axis for plot
L1, L2	Integration time measure interval
S4(41)=0	
Q1=1	1: Normal measurement 2: Blank measurement
Q2=1	1: Increasing substrate 2: Decreasing substrate

G1=1	Plot transformation
G2=1	Range to be plotted
W9=100	Delay between strings in sending
I9$= " "	User initials

After the local variables are defined and initialized, all stored data in the HP 8451 memory is erased and the display is selected as the graphic output device. The common data area flag C(8) is tested and the 'Autost' overlay is CHAINed if it is not set to 1. After that the DACOM Menu is shown.

4.4.2 DACOM Menu

The following screen is shown when this menu is active:

```
                  Dacom Menu
   Please select a softkey:



   D Var     D Edit
   Main M    Dacom     Content   Plot M
```

Function of Softkeys :

Softkey #1	activates the Edit Header Menu. Another function of this menu is averaging of blanks (GOTO 500).
Softkey #2	activates the Data Transfer Menu. This can only be done when the connection to the host is made and the data transfer program on the host is running (GOTO 2500).
Softkey #3	is used to show the derivatives of diskettes in drives :D700 :D701 and :D710 (line 1600). The directory is shown by the CAT command. Files can be

	erased (PURGE H1$ on line 1630) or the directory can be PACKed (line 1620).
Softkey #4	activates the Plot Menu (GOTO 7000).
Softkey #6	activates the Edit Data Menu (GOTO 2000).
Softkey #7	activates the Main Menu. To do that, the 'Autost' overlay is CHAINed on line 810.

4.4.3 Edit Header Menu

```
Duration      1     1  and 3.2    8   sec
Range    1   from      548     to   552
Range    2   from      416     to   420
Int. Ref.    from      580     to   586
Cytochrome Oxidase         20  nM   (E)
Cytochrome c              200  uM   (S)
[S] Assay    30 /     Stdev   3%
No. of Rep.    3 / No. of Points   20
dEps 1 - 95.9  dEps 2 - 69  1/mM  cm
Mixed Row

StorB     ReadB     CopyF     Avg Bl
Exit      ReadV     Dacom     Comme
```

Function of the Softkeys :

Softkey #1	exits to the Dacom Menu
Softkey #2	prompts for a file name of a data file and reads the header procedure on line 700.
Softkey #3	activates the Data Transfer Menu (GOTO 2500).
Softkey #4	is used to edit the comment (T1$, line 780).
Softkey #5	stores the content of the memory in a blank file (lines 1500ff). A name with 9 characters has to be entered on line

1500. The file is created, opened, the information is stored and the file is closed again on lines 1510-1590.

Softkey #6	reads a blank file into memory. Again a 9 character file name has to be entered.
Softkey #7	copies files line 900.
Softkey #8	averages up to five blank (A1 on line 3520). The loop from line 3550 to 3580 accepts A1 times a 9 character file name. The headers of the blank files are compared on lines 3610-3640 and the data is added up in the T1() and T2() arrays. The data is then divided by A1 and the store blank function is used to store the averaged blank.

4.4.3 Edit Data Menu

```
 ***********  Edit Data Menu  ************
   No.     Range 1      Range 2     [Substr]
    0      +.635683     +.935534     0.00
    1     +3.227895    +1.446329     0.47
    2     +8.216934    +7.938393     0.69
    3    +11.503677   +11.339026     0.92
    4    +13.528363   +13.290385     1.16
    5    +16.496043   +16.433786     1.42
    6    +18.816847   +19.833377     2.71
    7    +22.015744   +22.338385     2.03
    8    +24.843676   +24.907806     2.40
    9    +26.413485   +26.835763     2.82

   Incol      ReadD      StorD      PlotM
   Exit       NextP      FormP      Input
```

Softkey #1	exits to the DACOM Menu GOTO 1000.

Softkey #2	is used to show the next part of the data (line 2300). To do that, the page counter R1 is incremented. It is set to 1 if it is larger than 4.
Softkey #3	is used to show the former page of the date (line 2310). The page counter R1 is decremented and set to zero if it is negative.
Softkey #4	is used to enter the value of range 1 (column = 1, T1(K)), the value of range 2 (column 2, Ts(K), and all the substrate concentrations (volume = 3, S4(K)).
Softkey #5	is used to enter all values of range 1 (volume 1, T1(K)), all values of range 2 (volume 2, T2(K)), and all the substrate concentrations (volume = 3, S4(K)). The data is input by means of the loop from line 2460 to line 2490. The FOR-NEXT loop stops when the maximal number of data points is reached (T(5)).
Softkey #6	reads a file by means of the READ_KIN procedure on line 5000.
Softkey #7	stores the memory content to a file by means of the STORE_KIN procedure on line 5500.
Softkey #8	activates the Plot Menu (GOTO 7000).

4.4.4 Data Transfer Menu (lines 2500ff)

The first time this menu is activated, the user initials I9$ have to be entered. The menu appears as follows:

Data Transfer

Pausing between lines 200 msec

Exit Abort Master Wait

Functions of the Softkeys:
The connection to the host has to be established after the first three lines have been shown. This is done by the SETUP_TRANSFER procedure. If the connection is established, the rest of the menu and the softkeys are shown.

Softkey #1	exits to the Dacom Menu (GOTO 1000).
Softkey #2	stops a running data transfer and returns to the Data Transfer Menu (GOTO 2500).
Softkey #3	starts a data transfer. The directory of the data diskette is shown and a file name (D$) is expected to read a file by means of the READ_KIN2 procedure. The file name is truncated to a length of 10 characters, no file is read if *'memory'* is typed in. On line 2620 a timeout statement prevents bus hangups. The file name of the file on the host disk has to be entered B8$. The file name is truncated to 8 characters or to 10 characters if a drive (a: or b:) is included. The file on the host is opened by means of the OPEN command with the parameters: <file name> and <extension>. The extension is composed of the character "K" and the user initials. The

command string is sent to the host by means of the OUTPUT Z$ USING "30A"; H1$ and the TRANSFER Z$ TO 10 INTR statements. The host confirmation is received on line 2740 (ENTER 10, H1$). The file is sent by means of the SEND_KIN procedure. This procedure sends a CLOSE command before it ends. Upon finishing, the Data Transfer Menu is shown again. If an error occurs during the data transfer, the message on line 2780 is shown and the data transfer is aborted.

Softkey #4	is used to change the waiting time between strings (line 3300 W9).
Softkey #5	shows the Edit Header Menu (GOTO 500).

ERROR_RECOVER procedure (lines 2900ff)

This procedure checks the error lines and error numbers. If the error number is 67 and the error line is 5060 the message *'File <name> not found'* is shown. If the error number is 130 the message *'Diskette <volume> is not found'* and if the error number is 129 the message *'Diskette damaged'* are shown. If an error with a different error number occurs the message *'Error during data transfer'* is shown for two seconds and the current data transfer is aborted.

SETUP_TRANSFER procedure (lines 3000ff)

The RS 232 data transfer protocol is setup as follows:

```
CONTROL 10,2  ;  7      Modem lines are set
CONTROL 10,3  ; 15      Baud rate 9600
CONTROL 10,4  ;  7      8 bits 2 stop bits no parity
CONTROL 10,5  ; 48      Select handshake
CONTROL 10,11 ;192      XON/XOFF handshake
CONTROL 10,14 ; 19       XON/XOFF handshake
CONTROL 10,15 ; 17       XON/XOFF handshake
```

A timeout is set before the message is PRINTed to the host (line 3120). The reply is received by the ENTER 10; H1$. Data transfer is stopped if no reply or if a different reply is received (line 3140). If a timeout occurs during the setup, the following screen is shown:

```
'Please check:
a) if data transfer program on host is running
b) if RS-232 cable is plugged in correctly
c) if handshake protocol is correct
   (see manual)
Press CONT key
```

Pressing CONT shows the Dacom Menu from where another attempt can be started.

PLOTTING procedure (lines 4200ff)

This procedure is called with the parameter h2$ set to "1" or to "0". If this flag is set, the procedure PLOT_AXES is called to draw the axes on the plotter. Otherwise the axes are just shown on the screen with the DIS_AXES procedure. The code on lines 4280 to 4350 performs the mathematical transformation selected in the Plot Menu. If range 1 is selected the data is plotted with an asterix (*). If range 2 is selected the data is plotted with a plus (+).

SEND_KIN procedure (lines 4400ff)

The I/O buffer is cleared and a first header string is sent. It contains series D1$, number D2, date D3$, flags D4$, name of enzyme E1$, name of substrate E2$, and a comment T1$. Three more lines of header information is sent (see data structures). These entries are sent in fixed format ASCII (lines 4440 to 4460). Next the data is sent in the following formate:

"#,A,DDD,18A,20A,20A,20A,64A"
"A00120. Feb.1988..............ADH... ....ETHANOL...
Comment.."
After the last data point of the turnover number vs. substrate concentration data is sent the file is closed for data types different form 3 (line 4520). Otherwise OD vs. time data is read T(5) times from diskette S1(1) and sent in blocks of 16 values in fixed format.
"#,SZ.4DE,SZ.4DE,SZ.4DE"
"#,SZ.5D"
READ_KIN procedure (line 5000ff)
This procedure shows the directory of the data diskette and inputs a file name (D$). It then calls the READ_KIN2 procedure to read the content of the file.
READ_KIN2 procedure (lines 5050ff)
The file D$ is opened and the header is read (line 5070). Then the turnover number versus substrate concentration list is read (lines 5080-5120). The file is closed if the data type is different from 3. Otherwise it is left open so that the absorption versus time data can be read.
STORE_KIN procedure (lines 5500ff)
After the directory of the data diskette has been shown, a file name has to be entered D$. A file with a length of 10 records is opened and the header and data are stored. The file is closed before the procedure ends.
PLOT_AXES procedure (lines 6000ff)
This is the same procedure as in the 'Autost' overlay with the difference that the commands are not stored in a file but sent to the plotter directly.
4.4.5 Plot Menu

```
    **************  DACOM  ****************
                  PLOT MENU
```

```
Title of Plot
X-Axis     0      10      2      0
X-Label    .............................
Y-Axis     0      10      2      0
Y-Label    .............................
TN vs. [Substr.] Plot
Range 1
           X-Axis    Y-Axis    Pl Type
Range      StorD     PlotD     Page 1

Title      EditD
Exit       ReadD     Content   Page 2
```

First set of softkeys:

Softkey #1	exits to the DACOM Menu (GOTO 1000)
Softkey #2	reads a data file by means of the READ_KIN procedure.
Softkey #3	shows the directories of the diskettes.
Softkey #4	selects the second set of softkeys (GOTO 7200).
Softkey #5	is used to input the title of the plot (E5$).
Softkey #6	activates the Edit Data Menu (GOTO 2000).

Second set of softkeys:

Softkey #1	selects either range 1 or range 2 (G2).
Softkey #2	stores the data by means of the STORE_KIN procedure.
Softkey #3	plots the data by means of the plotting procedure.
Softkey #4	selects the first set of softkeys (GOTO 7100).
Softkey #6	is used to enter the X-axis variables and the X-axis label.

Softkey #7	is used to enter the Y-axis variables and the Y-axis label.
Softkey #8	is used to select among four transformations (G1)
	1: Tn vs. substrate
	2: Lineweaver Burke Plot
	3: Eadie Hofstee Plot
	4: Hanes Wolfe Plot.

DIS_AXES procedure (lines 8500ff)

This procedure has been described in the KIN2 overlay.

PRINT_HEADER procedure (lines 9500ff)

A similar procedure has been described in the 'KIN2' overlay.

4.5 Initiation Overlay 'ASINIT'

The 'ASINIT' overlay is executed when the common data area has already been initalized (C(8)=1) and when the initialization has not yet been performed (C(15) # 1, lines 250 to 490). The softkey #1 is used to stop the downloading process and to exit back to the 'Autost' overlay (lines 9000 and 9100).

On line 500-590 the following parameters are downloaded for each syringe

W (I + 5)

W (I + 15)

W (I + 10)

The commands are sent to the ASSAYOMATE by means of the SEND_CMD procedure (Has been described in the 'KIN2' overlay).

- SETSF procedure (lines 7100ff)
- CONVERT_HEX procedure (lines 7200ff)
- RECEIVE_FILL procedure (lines 7700ff)

These procedures have been described in the measure overlay on the same program lines.

chapter 5
List of Procedures

5.1 Alphabetical Lists

5.1.1 Start Overlay

AXES_DELOD PROCEDURE	5200ff
AXES_OD PROCEDURE	5100ff
AXES_TNS PROCEDURE	5000ff
CHECK_DIODES PROCEDURE	4800ff
CHECK_PARAMETER PROCEDURE	4700ff
DISPLAY_INHALT PROCEDURE	2000ff
EDIT_INHALT PROCEDURE	2200ff
ERROR_RECOVER PROCEDURE	9000ff
READ_PARAMETER PROCEDURE	4400ff
STORE_AXES PROCEDURE	6000ff

5.1.2 Measure Overlays

ASSAY_PROCEDURE	3500ff
AUTOKIN_PROCEDURE	2000ff
AVERAGE_VEL PROCEDURE	3750ff
BUILD_QUEUE PROCEDURE	900ff
CALCTN PROCEDURE	8200ff
CHECK_AVG PROCEDURE	3800ff
CONVERT_HEX PROCEDURE	7200ff
DATA_HEADER PROCEDURE	3700ff
DEBUG_MENU PROCEDURE	4450ff
DERIVATIVE PROCEDURE	4600ff
DIS_AXES PROCEDURE	8500ff
EDIT_COMMENT PROCEDURE	1700ff
ERROR_RECOVER PROCEDURE	1660ff
FUBED PROCEDURE	8000ff
GETABS PROCEDURE	4400ff
LINCHECK PROCEDURE	3300ff
LINREG PROCEDURE	6000ff
MEAS_REFERENCE PROCEDURE	1900ff
OPEN_DFILE PROCEDURE	5500ff

OUTKIN PROCEDURE	4200ff
PLOT_DATA PROCEDURE	6500ff
PRINT_HEADER PROCEDURE	9500ff
RECEIVE_FILL PROCEDURE	7700ff
SEND_CMD PROCEDURE	7000ff
SETSF PROCEDURE	7100ff
SETUP_KIN PROCEDURE	3600ff
SFSET PROCEDURE	7900ff
SINGLE_ASSAY PROCEDURE	4800ff
STORE_BLANK PROCEDURE	1540f
SUBTRACT_BLANK PROCEDURE	1800ff
SYRINGE_CONTROL PROCEDURE	7250ff
WASH PROCEDURE	7500ff
5.1.3 Testmeasure Overlay	
ERROR_RECOVER PROCEDURE	1660ff
FUBED PROCEDURE	8000ff
KINCONST PROCEDURE	4900ff
MEAS_REFERENCE PROCEDURE	1900ff
MEASURE_PRODUCT PROCEDURE	2500ff
MEASURE_SUBSTRATE PROCEDURE	2000ff
RECEIVE_FILL PROCEDURE	7700ff
REFERENCE PROCEDURE	1900ff
SEND_CMD PROCEDURE	7000ff
SEND_KIN PROCEDURE	6200ff
SERIAL_ASSAY PROCEDURE	5010ff
SETSF PROCEDURE	7100ff
SFSET PROCEDURE	7900ff
SSOPTION PROCEDURE	6050ff
STABILIZE PROCEDURE	3000ff
STORE_KIN PROCEDURE	6100ff
SYRINGE_CONTROL PROCEDURE	7250ff
TEST_ASSAY PROCEDURE	5000ff
WASH PROCEDURE	7500ff

APPENDIX C
The Michfit Software Documentation
By Bruno Michel
Table Of Contents

Chapter 1
Introduction

The ASSAYOMATE is able to measure a lot of assays within a short time. Especially, when the entire time course of the assays is stored, a large quantitiy of data has to be processed. For this reason it is necessary to have a possibility to automatically process these data. Typically, spreadsheets are used for the purpose of calculating result of data arrays and for their graphical presentation. The commercial spreadsheets are most efficient when a few data arrays have to be typed in and presented. When it comes to big amounts of data which are stored in digital form, the import of these file is quite time consuming. Managing large numbers of files with their small cryptic file names is getting complicated and difficult.
In this case it is advantageous to have a powerful devoted software for the processing of the data.

Another advantage of a tailored software is the possibility to include the data acquistion into the same package.
A workfile containing 100 datafiles together with a lot of header information makes file management much easier. In addition to that the access to files within the workfile is much faster than an access to DOS files would be. Workfiles are used to group together sets of files. Several workfiles can be used if a larger number of files has to be stored.
Files can be exchanged between workfiles by using plotfiles. This filetype is normally used to store plots as they appear on the screen in a file.
The MICHFIT software uses several overlays to perform subfunctions.

chapter 2
System Concept

The MICHFIT software was designed with the intend to minimalize the costs of software maintenance. This was achieved by removing screen masks and program messages from the source code. Screen masks are stored in specialized screenfiles which are accessed from the program by random file input/output commands. Program messages are stored in system files. For each menu an overlay of messages is read in. Similarly help screens are read in from a specialized file by the program. This structure has the advantage that the same program can, without any modification, run with different languages. In addition to that, the user interface can be customized by an experienced user with no need to change the source code.

Mathematical models used for enzyme kinetics change quite often. For this reason customization of fit equations is essential. This was achieved by a model interpreter which accepts and interprets ASCII strings from the system files to be used for fits thereafter.

Chapter 3
SYSTEM SOFTWARE

The high level languages Microsoft complied BASIC or Microsoft PASCAL implement only a small set of commands. All other commands or functions have to be implemented with CALLS to assembly language subroutines that are linked to the object modules. Among the functions that had to be implemented are:

- graphic functions
- touchscreen functions
- access to system clock
- read disc directories
- input/output routines (RS-232)
- multiple screens

Some functions can be executed by printing ESCAPE sequences to the terminal, but at a very slow speed. The HP-150 offers a built-in library of functions accessible by assembly language calls, the AGIOS functions.

The alphanumeric and graphics input/output subsystem (AGIOS) is a set of system functions that are used to access the alphanumeric and graphic intrinsics of the HP-150. These function calls perform text and graphic mode operations on the display, control the keyboard, define application softkeys and perform touchscreen operations. AGIOS provides a five times faster access

to the terminal features than comparable escape sequences. In addition AGIOS provides functions for transferring a block of data to the display.
Each AIOS or GIOS function is identified by a 16 bit function code. The function code and additional parameters have to be stored in a command buffer (see system software). Then the processor registers have to be set up, and the function is executed by an interrupt to the operating system.
The HP-150 provides a set of data communication I/O functions available in MS-DOS. The *'data comm'* functions are accessed in the same way as the AGIOS calls (by a command buffer).
The performance of the MICHFIT software was considerably inproved by two routines that store and recall the content of the alphanumeric screen from a mass storage medium:
The STOSCR (#) function is used to store the contents of the alphanumeric screen to random access file on a mass storage medium, preferably a RAM disc or a hard disc. With the RECSCR (#) function stored screens are recalled from a random access file that has already been opened. Only the first 21 lines of the screen are stored. The only limitation to the number of screens that can be stored, is the storage space and the speed of access.

3.1 Interrupts to the Operating System

Each AIOS or GIOS function is identified by a 16 bit function code.The function code and additional parameters have to be stored in a command buffer.

```
IN BUF      DW        1             ;function code
            DB        14            ;lower right column
            DB        16            ;lower right row
            DB        10              ;upper left column
            DB        15              ;upper left row
            DW        0FFFFH
            DW        0FFFFH
```

Then the processor registers have to be set up. The DS register has to contain the buffer's data segment.

```
DEFA        MOV       AX,4403H            ;I/O control
write
            MOV       BX,1                ;console handle
            MOV       CX,10               ;buffer length
            MOV       DX,OFFSET IN_BUF    ;offset address
            INT       21H                 ;call function
            RET       return              ;return
```

And the function is executed by an interrupt to the operating system.

The HP-150 provides a set of data communication I/O functions available in MS-DOS. The *'data comm'* functions are accessed in the same way as the AGIOS calls (by command buffer).

3.1.1 Control of Touchscreen

The function FNTOUCH defines a field on the screen of the HP-150 with the upper left corner (row%,col%) and the size (rowinc%,colinc%), that generates the reponse (resp$) when it is touched. Touching such a field is equivalent to pressing a key on the keyboard. The function OFFTOUCH removes all defined touchfields from the screen.

The alpha numeric screen can be cleared (CLS), switched on (ALPHA) and switched off (ALPHAOFF). The cursor can be located anywhere on the screen with the function CURS (col%,row%). One line of information can be written to the alpha memory or read from the alpha memory with the function READLI (H1) and DISLI (HS).

3.1.2 Acess to System Clock and Disc Directory

Among the functions most necessary for a user-friendly software are the access to the disc directories. This is especially important when a large number of files from different users have been stored on a hard disc. The functions SUCHF and SUCHN are used to select only files that belong to a specific task. These files can be accessed subsequently simply by touching the fields in the INPUT Menu.

Two functions are provided to read in the internal clock:
The function DATUM (year%,day%,dayofweek%) returns three 16 bit integers. The first integer is the year. The higher 8 bits of the second integer is the month, and the lower 8 bits the day of this month. The third integer returns the hour of the week (1-7) starting from Sunday. The function ZEIT (minute%, second%) returns the hour of the day (0-24) in the upper half of the first integer and the minutes in the lower half. The upper half of the second integer contains the seconds and the lower half contains the hundreth of seconds.

3.1.3 Data Transfer

Function POPEN (handle%) opens the serial port. It can then be treated as a character device from DOS. Function PCLOSE (handle%) closes the serial port. Function INSTAT (status%) checks if a byte is ready for input from the serial port. Function OUTSTAT (status%) checks if the serial port is ready to accept a byte.

3.1.4 Access to Graphic Functions

The graphic screen as well as the alpha screen can be controlled independently. The HP-150 uses a monochrome graphic screen with a resolution of 640 x 390 pixels.

The functions switching the graphic screen on and off are GRAPH and GRAPHOFF. All entries of the graphic screen are removed by the GCLEAR function. Similarly to the alpha screen, several routines are used to control the graphic cursor. GCURS turns the graphic cursor on and GCURSOFF removes it. It can be moved absolutely to a pixel MGCURS (X%,Y%) where X has to be in the range of 0 to 640 and Y has to be in the range of 0 to 390. Lines are drawn using the functions PLOTD (X%,Y%) and PLOTU (X%,Y%). X% and Y% have to be in the same range as in the case of the graphic cursor. Lines can be drawn with different line types (0-10; see HP-150 manual). It is possible to write text to the graphic screen of the HP-150. The size, the direction, and the slant can be defined with the functions SIZE(X%), DIR(X%), and SLANT(S%) respectively.

3.2 Multiple Screens

The STOSCR (#) function calls 22 times a subfunction which transfers the content of one line of the alphanumeric screen to be the file I/O buffer. Similarly, the RECSCR (#) function calls 22 times a subfunction which transfers 80 bytes from the file I/O buffer into the alpha memory.

Multiple screen that can be read from and written to screen files offer the advantage of usage and cost effective software maintenance. Another important aspect of this approach is the much better performance of the software. The reason for the better performance of this software is the smaller need for computing power, because screens need not be separated each time the menu is shown. At the time a particular menu is called. The screen is read from the mass storage and only parts that have been changed since the menu was left are actually updated. The difference is most pronounced when a lot of floating point numbers appear on the same screen.

Because the screen update is performed in a multitasking environement (time-sharing) and keyboard input are processed with a higher priority than screen updates. The response time due to user commands is much better. In many applications the routines servicing one menu are kept together in one program overlay. If the user decides to change to another menu the overlay has to be loaded and initialized before the menu can be shown.

With multiple screens, however, the new menu is called and displayed by the former overlay before the new overlay is loaded. The only thing the new overlay has to do, is checking if some updates have to be performed.

It is obvious, that a program offering internal multitasking and multiple screens is more complex to be written. But the higher initial cost are compensated by the lower maintenance costs.

In recent years the computing power of personal computers has risen constantly. It might seem if increasing the performance of a software is a waste of time because it runs faster on a faster computer anyway. But the complexity of programs has risen parallel to computing power. The appetite for computing power will increase significantly when multitasking operating systems have become standard in PC's. In this case having a software with a smaller need for computing power is a big advantage.

The multiple screen design even profits more if it runs on a computer that uses channel architecture. This is because many simple datatransfers from and to memory including alpha and graphic memory can be done by a DMA chip simultaneously. That means the CPU does not need to spend time on reading a screen mask from disc to the alpha memory and even if an overlay is loaded at the same time the speed of the datatransfer is not affected. In conclusion the design used in the MICHFIT software is a good investment for the future.

Chapter 4
USER SOFTWARE

4.1 Principle of Operation

The MS-DOS operating system (version 2.11) is a single task single user system. The file access is quite slow, especially to a hard disc when many files have been created and deleted. The reason is the lack of a reorganization task for a hard disc and a poor organization of the directory:

The quickest access is achieved to files that are created shortly after the initiation of the disc and that are never deleted. As the program had to be split into several overlays that are loaded into memory one after the other, optimization of disc access was essential.

The overlays use a main procedure that acitivates the different subfunctions of the program one after the other. Single tasks of these subfunctions are very short so that a pseudo real time performance is achieved. The subfunctions are:

- screen update
- keyboard input
- background plot (from file)
- serial interface input (to file)

Before a menu is left, the content of the alphanumeric screen is stored in a file. When the same menu is again selected, the screen is recalled from the file. Then, the new functions of the softkeys are labelled. After these two tasks control is given back to the main procedure. This organization is especially useful if another overlay has to be loaded. In this case the new menu is displayed before the actual program is ready. The user is able to study the menu while the computer is still working in the background. Key presses are saved and executed as soon as the overlay is ready (max. 5 seconds). The subroutine checks the field of the masks (20) and changes them if necessary. As the menus work on different tasks, modifications of the content of one menu has no effect on other menus. Key presses can be detected in two modes:
- command mode
- interpreter mode

In the command mode each byte is executed immediately. In the interpreter mode the execution starts only after a (cr) byte has been received (see Interpreter Chapter).

4.2 Main Overlay

The MICHFIT software processes files generated by the BITRATE, the MICHKIN, and the HP 8451 spectrophotometer software. Data from other sources can be received via a RS-232 interface. It was written around a workfile, an array of 100 records stored in two files on the hard disc. These records are used as operands of mathematical operations or as sources for direct curve fits and graphical presentations. The main program has seven major functions:

- initialize COMMON data area
- read directories
- edit menu masks
- edit system files
- activate workfiles
- call overlays

The MICHFIT software is started by the main program (HA.EXE). The user has to select one account (1-20) and has to enter a six character password. If the password is enterd correctly, the initiation routine is started. First, several parameters defining the configuration are read from the main system file DEFAULT.HSY. Second, the common variable are initialized and the default values of the plotsize are read from the system file USERXY.HSY. Third, all data files on the disc a:, b:, and c: that belong to the user are read, sorted according to file type and stored in a random access file on the fastest mass storage device. After the initiation, the Main Menu is displayed. The Main Menu offers functions for modifications of system files and screen masks.

Moreover, it is used to initialize the system files and the workfiles for new users. Most of the time, the Main Menu is used to select among 4 different types of applications:

- processing of kinetic data
- processing of binding data
- processing of spectra
- data transfer

The first three data processing subroutines perform very similar tasks on different types of data. The only differences are the output transformations. The spectra processing application uses a different workfile than the other two subroutines. In the case of spectra only

Y-values are stored, the X-values have to be spaced evenly so that only X-min, X-max and delta X have to be stored.

4.2.1 Screen Mask Editor

The screen mask editor is a page oriented editor. It is preset to a fixed page length of 20 lines with 80 characters per line. It reads the pages with a fixed offset of n times 1600 bytes from the start of the random access file. When it is called it is assigned to one of the screen mask files. To do this, the user has to select an application *'Please select application: 1=Binding 2=Spectra 3=Kinetics'*. It then clears the screen and displays a set of softkeys:

f1	=	Recall Screen
f3	=	Read Default
f4	=	Store Screen
f5	=	Other Application
f8	=	End Edit

The user is asked to select one of the softkeys or to start editing the screen. After having written a new screen or after having modified an existing screen, it can be stored to a specified location. After the user has pressed softkey #4 the editor prompts: *'Please enter screen number in the range from 1 to 20'*. Depending on the number, the content of the screen is written to a different location in the file.

Screen	1	Input Menu
Screen	2	Output Menu
Screen	3	Plotsize Menu
Screen	4	Fit Menu
Screen	5	Data Handling Menu F 1 - 17

Screen 6	Data Handling Menu F 18 - 34
Screen 7	Data Handling Menu F 35 - 51
Screen 8	Data Handling Menu F 52 - 69
Screen 9	Data Handling Menu F 70 - 87
Screen 10	Data Handling Menu F 88 - 99
Screen 11	Help Screen Input Menu
Screen 12	Help Screen Output Menu
Screen 13	Help Screen Plotsize Menu
Screen 14	Help Screen Fit Menu
Screen 15	Help Screen Data Handling Menu 1
Screen 16	Help Screen Data Handling Menu 2
Screen 17	Help Screen Data Handling Menu 3
Screen 18	Help Screen Data Handling Menu 4
Screen 19	Help Screen Data Handling Menu 5
Screen 20	Help Screen Data Handling Menu 6

4.2.2 System File Editor

System files contain all messages displayed to the user by the MICHFIT software. These messages can be changed by using the system file editor. This small program reads a set of 10 prompts and displays them on the screen where they can be modified. There are different types of system files.

a) The main system file (<username>.HA)

b) The application specific system files (<username>.KSY, <username>.SSY a.s.o.)

Upon calling the editor the name of the file has to be selected. Then the editor prompts: *'Please select a page'*. It then accepts integer inputs from 1 to 10. As mentioned before one screen (or page) holds 10 messages. They can be accessed by typing 1-10 when the following prompt appears or the command line: *'Actual page 6 please select a line or 999 = exit, 888 = store or 777 = new page'*.

As can be seen in the prompt a modified set of 10 messages is stored by typing 888.
Pages above 10 have a different function, they hold the information that is necessary for the definition of the touchscreen fields. There are two groups of such fields:

a) single touchfields
b) touchfield arrays

The single fields are defined by two integers. The first defines the upper left corner, and the second defines the lower right corner. The integer number is split into two parts. The lower significant two digits hold the column position (0-80) and the upper two digits hold the new position (0-47).
Touchfields are generated until a 0 is encountered. The field arrays are defined as single fields but first a row of fields and then the leftmost column of fields is defined. Again generation of fields is stopped when a 0 is encountered. (More information in detection of touchfields)

4.2.3 Startup Routines HA.BAS/HA.EXE

Before the MICHFIT software can be started the user has to enter an account number and a password (290-340). Having a controlled access to the software has two advantages: The data of a particular user cannot be accessed by unauthorized persons. A superuser is responsible for opening accounts (line 10000 -10100) and for removal for unused accounts from the mass storage device. The maximal number of accounts has to be limited because overfilling of the mass storage causes less of performance of the operating system. After the user has entered the password properly, a series of ANSI escape sequences is printed to the terminal to ensure a constant environment for the software (lines 400 - 490). Then the date is read from the system clock with the DATUM(year%, day%, wday%) call (see System Software). It is then converted into an ASCII string and stored in the common data area. The copyright screen is shown and the user is asked to insert additional data diskettes. On lines 600-990 the common variables are initialized. The default values of the common variables can be read from a file (not yet implemented lines 350 - 390) so that these values can also be edited by the user with the system file editor. In order to reduce the number of access to the disc directories, the content of all directories are stored in a random access file (lines 9000 - 9190). To do this, the directories are read with the SUCHF and SUCHN functions. Then only files which belong to the current user are selected and stored.

After the first initialization procedure the Main Menu is shown (lines 1000 ff). There the user can select among different applications. In this documentation only the kinetic application is described. If the application has been activated, the file buffers for system file, workfile, screen file, and directory file are opened (lines 4400 ff). If the account has never been used before, the system files and screen files have to be copied from the DEFAULT files (subroutine on line 4100 ff).

Before the HA program is left with the CHAIN command on line 1990, a garbage collection is started with the command X=FREE (""). Before that, all unnecessary strings are cleared. Performing small garbage collections at several places in the software saves memory space and keep delays short. If no garbage collections are performed intentionally they are called automatically when no free space is left in memory. These garbage collections usually happen in bad moments and can last very long.

4.2.4 Terminating Application (lines 850 ff)

When the MICHFIT application switches to another menu that needs loading of an new overlay, the RUNNING% variable is set to zero and the IFORT variable is set to zero. In this case, the subroutine supporting the other menu is activated with the CHAIN fort$ command on line 840. If, however, the IFORT pointer is zero too, program execution resumes with line 850. Here most file buffers are closed. As a next step the menu update pointers and the workfile update pointers are written to disc (lines 950 - 965). Before the program stops the user has to answer the question *'Save current status (y/n)?'*. If the user enters *'n'* the current status file is erased and the software starts with default values the next time it is called. If the user enters *'y'*, the current status file with the username and the extension *'.KST'* is opened on disc a:. All pointer values are written to this file (lines 975 - 982). The active plotsize values are written to the same file using the write plotfile subroutine (lines 8035 ff). After having closed the current status file, having erased the touchfields and having cleared the display the message *'MICHFIT software terminated'* is displayed and control is returned to the operating system.
Altenatively the save current status routine can be called by the 'EXIT MAIN MENU' function (softkey #1 in the INPUT Menu). In this case the MICHFIT application is stopped and the MAIN Menu is shown. From this menu

other applications can be started (e.g. spectra data handling, data transfer).

4.3 Data Handling Overlay

4.3.1 Local Variables

When the MICHFIT application is started, the IFORT variable is set to zero by the compiler. At the time of the first start the program branches from line 330 into the start subroutine at line 4100. This subroutine clears the screen, shows the copyright message and sets all system pointers to default values:

default = 0	no plot defaults have been read
anzeige = 1	page 1 of Data Handling Menu is shown
autoscale = 1	plots are scaled automatically
interpolate = 0	no additional points are generated in between data points
anzart = 1	amount of information shown in Output Menu
softkey = 1	the first softkey set is active in Data Handling Menu
teart = 1	output transformations are switched off
anzgem = 1	data files are read one at a time and not averaged
interpreter = 0	operation mode is direct command execution
offo% = 0	the Output Menu roll pointer is reset
comfile% = 0	macro mode is switched off
	pointer%(8) = 0
pointer%(20) = 0	menu update starts with Input Menu
interpk(i) = 0	no additional data points are

	generated in between data points
stift%(i) = 1	pen number one is used to draw all data lines or symbols

Next a subroutine which reads in the menu and workfile update pointers is called. This routine occupies the lines 18700 to 18730. After the update pointers have been read, they are cleared. This ensures that the menus are properly updated even if the computer stops because of a power failjure or an operating system error while the MICHFIT software is active. The routine at lines 18500 to 18640 reads in the messages for the prompts and other outputs to the screen. As mentioned above, these messages can be edited by the user by means of the system file editor. In the common data area these messages are stored in the FORMAT$ array.

iu(i) = read	update pointers are read from the system file
ip(i)	(this is performed in a subroutine at lines 18700 ff)

4.3.2 Menu Update

The menu update routines are identical for all menus. For this reason only the update for the Output Menu is described. The number of the Output Menu is 2. Line 710 in the main loop shows, that the update for the Output Menu starts at line 2025.

The lines from 2000 to 2020 are executed when the menu is recalled from the screen file. If the menu pointer is already 2, the next lines are skipped. Then all remaining touchscreens are cleared (OFFTOUCH), the alphanumeric screen is switched on (ALPHA), and special to the Output Menu, the maximal number of output curves is set to ten. Then the screen is read into the file buffer with the basic command get #14,2. The screen is

cleared and the content of the file buffer is copied into the alpha memory (RECSCR).
The menu pointer is set to 2 (IME=2) and the touchfields are generated using information stored in systemfiles. Next the softkeys are labelled (subroutine at line 8700). The text for the labels is read from the system file and passed to the subroutine in string hl$. The store screen pending flag is set (ip(39)=1) and the update pointer is set so that it points to the first display field. During each cycle of the main loop one small part of the active menu is checked and updated if necessary. If all parts have been updated the routine is left without any action (if pointer%(12)>30 then return).
The update pointer is incremented and the active part of the menu is checked if it needs update. Depending on the result of the check the parts are updated. The update procedures for the Output Menu are located on the lines 2200 until 23900.
The last entry into the command list is 6800. This procedure stores the modified screen back to the file. It is called, once the entire screen has been updated, or immediately when the menu is left.

```
Spool  Help   *** OUTPUT  **** 09. Feb. 88  ** User  :
Michel  *******
Mode          :   X-Y Output      Output to      :   Display
Automatic Scale  :   YES    Default Values  :   NO
Line interpolate :   NO     No. of Curves   :   2

Register File name   Symb/Pen Y-Max. [E]  [S]  From  To
..........................................................
27 Alcohol dehydrogen. 6 1   87.62  5.32   30   1    20
12  ADH with inhibitor 7 2   12.23 34.30   30   1    20
```

As in other menus, the comments are separated into three groups. The first group are functions assigned to a single touchfield. The second group are functions assigned to touchfield arrays. The third group of commands are activated by the softkeys.
Line 2400 is the entry point of the execute command subroutine. When it is called either the TFELD, the TARRAY, or the TSOFT flags are set to a value different from zero. The TFELD and TSOFT values are directly the numbers of the touchfields respectively the softkey. The TARRAY pointer indicates which array has been touched. To identify a single field two more values are passed: TCOL and TROW. The subroutines for the single fields are between lines 2440 and 2590.
The functions of the touchfields and of the softkeys are described in the Output Menu section.

4.3.3 Timesharing (main routine)
Once the overlay has been loaded, local variables and functions are initialized (lines 90 -285) and filebuffers are reassigned (lines 300 - 320). Then, depending on the pointer 'FORTS', program execution starts with different procedures:

FORTS	Task	Line and procedure name	
0	Activate Workfile	4100	START
	Show Input Menu	1000	INPUT_MENU
1	Show Input Menu	1000	INPUT_MENU
2	Show Output Menu	2000	OUTPUT_MENU
3	Show Data Handling Menu	3000	DAHA_MENU
4	Return from external overlay	4000	RETEXT

	and show Output Menu	2000 OUTPUT_MENU
5	Read System file overlay	18500 READ_SYSFILE
	and show Input Menu	1000 INPUT_MENU
6	Read System file overlay	18500 READ_SYSFILE
	and show Output Menu	2000 OUTPUT_ MENU
7	Read System file	18500 READ_SYSFILE
	and show Data Handling Menu	3000 DAHA_MENU
8	Store Plotfile	8000

WRITE_PLOTFILE

The main routine is a DO WHILE loop from line 500 to line 790. It executes several tasks in a pseudo multitasking manner.

- It monitors keypresses by the user and outputs data from the plotspool to the plotter (lines 5000ff)
- As a result of keypresses it takes actions
- And it updates the active menu screen.

Kepresses can be detected in two modes:

- In the direct mode each key corresponds to a function.
- In the interpreter mode an ASCII string is accummulated and analyzed.

Usually ASCII strings are entered directly by the user. They can, however, be read from a file (MACRO) and interpreted. The following lines describe the main routine :

500	while running%	Start of do while loop that runs indefinitely until running% is set to 0 by a function
505	while warnz% > 0	Displays warning messages that have occured then gosub 6300 during execution of last command

510	if comfile% > 0 then
	a$ = macro$(comfile%)
	comfile% = comfile%+1
	if comfile% > 21
	then comfile% = 0
		if the MICHFIT software runs in the macro mode the current macro is read into the command buffer a$ and the macropointer is incremented if the macropointer is > 21, the macromode is switched off.

515	if comfile% > 0 then gosub 4500	if the MICHFIT software runs in the macor mode the command in the buffer a% is interpreted and executed in a subroutine starting at 4500.
520	tshare = 10	Timeshare of the plotter is set at 10%
ileer = 1		
ilock = 0		unlock input
if comfile% = 0 and interpreter <>1 then gosub 500		if the MICHFIT software runs in the normal mode it checks if the user has pressed a key. At the same time it checks each ten cycles if the plotter needs more commands.

560	if ime <1 or
	ime > 6 then
	ime = 1
		Check if the active menu pointer is in the valid range (1-6)

565	if flag = 0 then 700	If no key has been pressed no command has to be executed
570	on ime gosub	Execute command depending on which menu is active
	1400	Input Menu commands
	2400	Output Menu commands
	6700	Data Handling Menu commands
	6700	Help Menu commands
	6950	

	17400	
700	if pointer% (20) <100 then pointer% (20) = +1	Test if the workfile headers have not yet been read
	zr% = pointer%20 gosub 9800	Call HEADER_READ procedure to read the header information for the next entry in the workfile
710	on ime gosub	Update Menu command depending on IME
	1025	INPUT_UPDATE
	2025	OUTPUT_UPDATE
	3025	DAHA_UPDATE
	990	-
	990	-
	990	-
790	WEND	End of DO WHILE LOOP
800	gosub 900	Clean string space
830	if ifort = 1	Read the overlay from disc
	pointer%(2) = idmax	Number of draw points
	pointer%(3) = comfile%	Macro mode on or off
	pointer%(4) = interpreter	Interpreter mode on or off
840	if ifort = 1 pointer%(19) = ime chain fort$	Read the overlay from disc and save pointer to active menu

After the menu and workfile udate pointers have been loaded, the START procedure searches for the current status file (line 4115). If it is not found the message: *'<username>.KST not found'* is displayed. Otherwise the current status file is opened and the pointer values from the last use of the software are loaded. In this case the message *'reading <username>.KST'* is displayed. Similarly to the TERMINATE procedure, a part of the READ_PLOTFILE procedure is used to read the plotsize variables. This routine is accessed without reading data (pg(8)=0, number of data curves = 0).

Next the graphic screen update pointers are cleared (because the graphic screen is empty) (pointer%(21)=0), and the zoomed mode is selected (pointer%(23)=0). The system file entry FORMAT$(52) is checked if it contains the command 'AUTO' if this is the case the MICHFIT application starts in the macro mode (interpreter = 1, comfile% = 1). Control is returned to the main program with the first macro line containing the system file entry FORMAT$(53). This can be used to load an autostart MAC or a demo.MAC file. (See command interpreter for more information).

4.3.4 Input Menu and Procedures

INPUT_MENU procedure (lines 1000ff)

Similar to the Output Menu, the screen mask is read from the screen file, the softkeys are labelled and the touchfields are set up. Once this has been done, control is returned to the main program (see menu update).

INPUT_UPDATE procedure (lines 1025ff)

If the Input Menu is active, the Main Menu calls the INPUT_UPDATE procedure once in every cycle. During each cell, one call of the display is updated. Here, a list of Input Menu update calls:

No.	function	line	row	col
ip(1)	source	1200	22	2
ip(2)	teart (filetype)	1205	64	2

ip(3)	anzgem number to average	1220	22	4
ip(4)	anzit number of data curves	1225	64	4
ip(5)	tque(i1) array of destination	1250		6
ip(6)	registers in workfile	1250		8
ip(7)	GET_DIR read directory	4200		
ip(8)	inhalt$() file names line 1	1300		12
ip(9)	inhalt$() file names line 2	1300		14
ip(10)	inhalt$() file names line 3	1300		16
ip(11)	inhalt$() file names line 4	1300		18
ip(12-18)	no function	1100		
ip(19)	clear command line	1390		21
ip(20)	store updated screen	6800		

```
Spool  Help *** INPUT MENU *** User: ** xxxxxxxxx  ***

Data source  :  Drive A    File type:      Kinetic data

No. to average  :  2        No to input:          2
                   1    2    3    4    5    6    7    8    9    10

Dest. register:    10   11

Dest. register:



*******************************************************

HORSCCA1*HORSCCA2*HORSCCA3 HORSCCA4 MODFCCB1MODFCCB2aao
```

INPUT_COMMAND procedure (lines 1400ff)
The INPUT_COMMAND procedure is called when a user input is detected while the Input Menu is active. The user input is converted by the INCONVERT routine. The

parameters that are passed to the INPUT_COMMAND procedure are:

TFELD

TARRAY TCOL TROW

TSOFT

The commands are divided into three groups:

a) the single touchfields

b) the touchfield arrays

c) the softkeys

If the TFELD parameter is set to a value between 1 and 10, one of the following single field commands is executed:

1	1460	switch to next input source (tein=tein+1)
		ip(1) update source field
		ip(7) get directory of new drive
2	1470	switch to next filetype (teart=teart+1)
		ip(2) update filetype field
		ip(7) get directory of new filetypes
3	1480	enter number to average anzgem=h1
		get input with INVAR procedure on line 3960
		ip(3) update number to average field
4	1490	enter number of curves anztit=h1
		ip(4) update number of curves field
		ip(5) update destination fields
		ip(6) (both line)
5	1500	enter destination registers as a block
		ip(5) update destination fields
6	1510	store Input Menu and show
7	1520	help screen
8	1530	command to plot spool
9	1540	no action
10	1550	no action

If the TARRAY parameter is set to 1 or 2 one entry in the array can be accessed with the TCOL and TROW parameters. Line 1600 selects either array 1 or 2.

Array 1: Destination Registers

The TCOL and TROW parameters are used to move the cursor to the appropriate field (lines 1610 and 1620). A question mark is placed in the field and an integer in the range from 1 to 99 is expected.

Array 2:   Directory entries

The TCOL and TROW parameters are used to move the cursor to a position just after the file name so that an asterix (*) can be printed (lines 1679). The file is identified by the TOUCHF parameter. This is used to copy the file names from the inhalt$() string into the file names$() strings which are passed to the input routines (see softkeys).

If the TSOFT parameter is set to a value in the range from 1 to 8 one of the softkey commands is executed (lines 1720 to 1860).

1   EXIT MAIN (line 1720)

The Input Menu screen is stored, the current status is saved and control is passed to the startup program.

2   RENAME FILE (line 1740)

A message is displayed on the command line. The INPUT_FILENAME procedure is called which waits until a touchfield is touched. If one of the file names has been touched, this is taken as the source file name. Otherwise the file name has to be typed in. The source procedure applies to the new name (line 1743). Lines 1743 and 1745 perform the rename operations.

3   EXECUTE INPUT (line 1750)

This function stores the Input Menu screen and passes control to the input overlay *'IKI'*.

4 ERASE FILE (line 1780)
A message is displayed on the command line (FORMAT$(12)) and a file name has to be entered (using INPUT_FILENAME procedure on line 1900). To prevent loss of data the file is renamed to 'RESCUE.SYS' instead of deleting it (line 1787). The file name in the internal directory is removed (subroutine on line 1950), and the modified internal directory is read (subroutine on line 4200). Then the update pointers are set and the function ends.

5 SHOW FILE (line 1800)
MICHFIT data files can be shown on the screen without reading the data. Again, this function uses the INPUT_FILENAME procedure to get the file name. The content of the file is shown on the screen. It can be printed as well on the line printer when a *'p'* is pressed. Hitting the *'a'* key aborts display and print immediately.

6 COPY FILE (line 1820)
Serial data files can be copied by using this function. Again, the INPUT_FILENAME procedure is used to input the source file name (line 1821). The same applies to the destination file name. A new entry is added to the internal directory if the file name does not exist already (line 1822). Lines 1823 and 1826 copy the file and set the update pointer.

7 DOS DIRECTORY (line 1840)
This function uses the BASIC files function to show the entire directory of the active drive.

8 END MICHFIT (line 1860)
The RUNNING% variable is cleared while IFORT remains zero. This stops the infinite main loop. Before control

is returned to the operating system the current status is saved.

INPUT_FILENAME procedure (lines 1900ff)

This procedure copies the file names from the internal buffer (inhalt$()) into h1$, if one of the file names touched. Deternatively, the file names can be typed in by the user if any other key is pressed. In this case the prompt *'Enter file name ? a: TEST ......XBM'* is shown and the TEST file name can be overwritten. If <endline> is pressed the modified name is input.

DEL_ENTRY procedure (lines 1950ff)

This procedure is called with the parameter i2 (number of entry), tein (drive), and teart (filetype). The file name is removed fro the selected directory and the following names are moved by one position.

ADD_ENTRY procedure (lines 1975ff)

This procedure checks if the file name passed in h2$ already exists in the internal directory. In this case the procedure ends with h1 set to 1 and the message *'File exist'* is displayed (line 1977). Otherwise, the new name is appended to the directory, and the number of files variable is incremented.

4.3.5 Output Menu and Procedures

OUTPUT_MENU procedure (lines 2000ff)

The lines from 2000 to 2020 are executed when the menu is recalled from the screen file. If the menu pointer is already 2, the next lines are skipped. Then all remaining touch fields are cleared (OFFTOUCH), the alphanumeric screen is switched on (ALPHA), and the maximal number of output curves is set to ten. Then, the screen is read into the file buffer with the basic command get #14,2. The screen is cleared and the

content of the file buffer is copied into the alpha memory (RECSCR).
The menu pointer is set to 2 (IME=2) and the touchfields are generated using information stored in systemfiles. Next the softkeys are labelled (subroutine at line 8700). The text for the labels is read from the system file and passed to the subroutine in string h1$. The store screen pending flag is set (ip(39)=1) and the update pointer is set so that it points to the first display field. During each cycle of the main loop one small part of the active menu is checked and updated if necessary. If all parts have been updated the routine is left without any action (if pointer & (12)>30 then return).
The update pointer is incremented and the active part of the menu is checked if it needs update. Depending on the result of the check the parts are updated. The updated routines for the output menu are located on the lines 2200 until 23900.
The last entry into the command list is 6800. This routine stores the modified screen back to the file. It is called once the entire screen has been updated or immediately when the menu is left.

OUTPUT_UPDATE procedure (lines 2025ff)
If the Onput Menu is active, the main program calls the this procedure once in every cycle. During each call, one cell of the display is updated. Here a list of Output Menu update calls:

No.	function	line	col	row
ip(21)	username	2200	27	0
ip(22)	type of trnsformation	2205	22	2
ip(23)	output destination	2210	64	2
ip(24)	automatic scale no/yes	2215	22	4

ip(25)	default values	2220	64	4
ip(26)	interpolate on/off	2225	22	6
ip(27)	ANZTIT number of curves	2230	64	6
ip(28)	output register 1	2295		10
ip(29)	output register 2	2295		12
ip(30)	output register 3	2295		14
ip(31)	output register 4	2295		16
ip(32)	output register 5	2295		18
ip(33-38)	no function	1100		
ip(39)	clear command line	2290		21
ip(40)	store updated screen	6800		

```
Spool  Help * OUTPUT ** 09. Feb. 88 ** User : Michel* *
Mode         :   X-Y Output     Output to     :  Display
Automatic Scale  :   YES        Default Values  :   NO
Line interpolate :  NO            No. of Curves :   2

Register Filename  Symb/Pen  Y-Max.  [E]  [S]  From  To
.......................................................
........................
27 Alcohol dehydrogen. 6 1   87.62    5.32  30    1   20
12  ADH with inhibitor 7 2   12.23   34.30  30    1   20
```

OUTPUT_COMMAND procedure (lines 1400ff)
This procedure is called when a user input is detected while the Output Menu is active. The user input is converted by the INCONVERT routine. The parameters that are passed to the OUTPUT_COMMAND procedure are:

TFELD
TARRAY TCOL TROW
TSOFT

The commands are divided into three groups:

a) the single touchfields

b) the touchfield arrays

c) the softkeys

If the TFELD parameter is set to a value between 1 and 10 one of the single field commands is executed:

1	2450	switch to next output transforamtion (tmod=tmod+1)
		ip(22) update mode field
		ip(41 - 50) output all data again
2	2460	switch to next output device (taus=taus+1)
		ip(23) update Output to field
		ip(62) same in Plotsize Menu
3	2470	switch autoscale on and off
		ip(24) update automatic scale field
4	2480	read default plot sizes
		execute DEFAULT procedure
		ip(25) update default field
5	2490	switch interpolate on and off
		ip(26) update interpolate field
6	2500	prompt for a number in the range from 1 to 10
		ip(27) update No. of curves field
		ip(28-32) and all output register fields
		ip(41-50) write data into output buffer
7	2510	prompt for a block of registers to be output
		ip(27+n) update n output registers
		ip(4-6) update destination registers in Input Menu
8	2520	prompt for the linetype of all selected registers
		ip(28-32) and update all output register fields

9	2530	prompt for a pen for all selected registers ip(28-32) and update all output register fields
10	2540	prompt for the maximal signal (Y. max) for all selected registers ip(28-32) update all output registers fields ip(41+n) write data into output buffer
11	2550	prompt for the enzyme concentration for all selected registers ip(28-32) update all output registers fields ip(41+n) write data into output buffer
12	2560	prompt for first data point for all selected registers ip(28-32) update all output registers fields ip(41+n) write data into output buffer
13	2570	prompt for last data point for all selected registers ip(28-32) update all output registers fields ip(41+n) write data into output buffer

If the TARRAY parameter is set to 1 or 2 one entry in the array can be accessed with the TCOL and TROW parameters. Line 2600 selects either array 1 or 2.

Array 1: output registers

The TCOL and TROW parameters are used to move the cursor to the appropriate field (lines 2610 to 2625). A question mark is placed in the field and one number is expected. The valid range of the number depends on the TCOL (lines 2640 to 2647). This array is used to change (Line, Pen, Y.Max, E, From, or To) of one output register.

If the TSOFT parameter is set to a value in the range from 1 to 8 one of the softkey commands is executed (lines 1720 to 1860).

1 ALPHA GRAPHICS (line 2720)

The alphanumeric screen is switched off and the graphic screen is displayed by means of the assembly language call (ALPHAOFF and GRAPH). The software pauses and waits for a user input. It then removes the graphic screen again and shows the alphanumeric screen with the assembly language calls (GRAPHOFF and ALPHA).

2 EXECUTE OUTPUT (line 2740)

Pressing this softkey executes the output according to the settings in the Output Menu. The procedure OUTPUT_TODEV is callled to load the output buffer with the data if this has not already been performed as a background task (by calling the OUTPUT_TODEV2 procedure from the menu update). Control is given to the plot overlay (running%=0 and fort$=plpgdr$+"pl". If an external program is needed to control another output device this is done by giving control to a program with the name stored in format$(60) in the system file.

3 CHANGE DISPLAY (line 2760)

The appearance of the Output Menu is changed by altering the ANZART flag (1 or 0).

4 CHANGE TRANSFORM. (line 2780)

Pressing this softkey prompts : *'Please enter new transformation: A = 1 / XRR'* and the cursor is placed under the first character of the X-transformation equation so that it can be modified (format$(25)). The same prompt is repeated for the Y-transformation (format$(26)).

5 ROLL UP DOWN (line 2800)
The roll% pointer is modified after the prompt *'roll (-5/+5) 1'* and the cursor is placed below the 1. The number in this field is added to the roll% pointer after an <End line> has been pressed. The roll% pointer is used to show the header information of the second set of 5 output registers (only five headers can be shown at a time).

6 PLOTSIZE + OUTPUT (line 2820)
This softkey performs the same function like softkey 2 with the difference that the Plotsize Menu is shown before the output is sent to the selected device. This is achieved by setting the PINIT variable to 1 instead of 2.

7 READ PLOTFILE (line 2840)
The procedure READ_PLOTFILE is called to input a file name and read the selected plotfile. Normally this function is performed using the Input Menu. There it is possible to get a list of stored plotfiles and to read in plotfiles by touching one of the fileds.

8 PRINT REGISTERS (line 2860)
After this softkey has been pressed, the prompt *'Please enter registers (from,to) '* is shown and the user can enter the range of workfile register haders to be printed on the line printer. If 0,100 is entered after the prompt, the entire header information of the active workfile is printed. The printed header information is the same as is displayed in the Data Handling Menu.

TRANSFORM procedure (lines 2900ff)
This procedure calls the TRANSFORM2 procedure to prepare the selected output register in the data buffer. The current Output Menu is saved in the screen file and other data in excess to the selected number of output registers (ANZTIT) are cleared.

TRANSFORM2 procedure (lines 2905ff)
This procedure reads the selected workfile register and performs the transformation TFORM) on it :

1 =	X - Y output	no transformation line 2940

2 =	double reciprocal	both axes are inverted line 2980
		xwert=1/Substratkonz
		ywert=1/Geschw

3 =	Eadie Hofstee	line 2960
		xwert=Geschw
		ywert=Geschw/Substratkonz

4 =	Hill Plot	both axes logarithmic line 2970
		xwert=log(Substratekonz)
		ywert=log(MaxGeschw - Geschw)

5 =	Hanes Wolfe	line 2950
		xwert=Substratkonz
		ywert=Geaschw/Substratkonz

6 =	Variable	user defined transformation

If the user defined transformation is selected (TMOD=6), the task is performed by the equation interpreter (procedure INTP_TRANSFORM, see interpreter chapter).

4.3.6 Data Handling Menu and Procedures
DAHA_MENU procedure (lines 3000ff)
The DAHA_MENU procedure reads one of the six content screens (ANZEIGE1 - ANZEIGE6). It then sets the update pointers for command line clear and for the equation update (line 3010). Next the touchfields are generated and the softkeys are labelled (lines 3015 and 3020).
DAHA_UPDATE procedure (lines 3025ff)

If the Data Handling Menu is active, the main program calls the DAHA_UPDATE procedure once in every cycle. During each call one line of the Data Handling Menu is updated. If the data handling update pointer (pointer%(13)) has a value between 1 and 17, one directory line is updated. The register number is calculated from the page number and the line offset (line 3290). If a line needs to be updated this is performed with the DIRECTORY_LINE-UPDATE procedure (lines 3300ff).
This procedure checks if the line lies within the screen boundaries and within the workfile (lines 3300-3310). It then moves the cursor to that line and performs a HEADER_READ call to the workfile. A part of the haeder information is used to build one line. The length of the parts is taken from the system file (FORMAT$(65)). The update pointer for that line is reset (in(zr%)=0) and the store screen flag (ip(60)) is set. Lines 3040 and 3042 are used to update the rest of the Data Handling Menu. If pointer%(13) is 21 the equation on the menu is updated (subroutine on line 3950). If pointer%(13) is 29 the command line is cleared (subroutine on line 3395). If pointer%(13) is 30 the modified screen is stored in the screen file if necessary (if the store screen flag (ip(60)) is set).
DAHA_COMMAND procedure (lines 3400ff)
The DAHA_COMMAND procedure is called when a user input is detected while the Data Handling Menu is active. Five parameters need to be passed to this procedure: TFELD, TARRAY, TCOL, TROW, TSOFT. Lines 3400 to 3440 activate single field routines, or softkey routines. No touchfield arrays need to be processed in the Data Handling Menu. Three single fields are used to input the parameters for the equation:

C = A * X + B * Y + Z

The operation is performed by pressing a softkey (see below).

The DAHA_COMMAND procedure supports two sets of softkeys. The active set is identified with the common variable SOFTK. Depending on this variable either the block at line 3700 or at line 3500 is called.

1 set of softkeys

The first set of softkeys provides the basic set of the workfile edit functions. The softkeys 1 thru 8 are identified with the TSOFT parameter. Line 3700 selects the appropriate subroutines for the softkeys:

1 CURVE FIT (line 3720). The softkey labels are changed and a message is shown. Otherwise the Curve Fit menu is not different from the Data Handling Menu. The Data Handling Menu is stored and control is given to the RKI (kinetic curve fit) program.

2 FORMER PAGE (line 3740). The current Data Handling Screen is saved and the former page is activated.If the first page has been reached, this command is ignored.

3 NEXT PAGE (line 3760). The current Data Handling Screen is saved and the next page is activated. If the last page has been reached, this command is ignored.

4 COPY REGISTER (line 3780) This function is used to copy a data set from location 1 to location 2 in the workfile. If softkey 4 is pressed the software prompts: 'Please enter the source and destination register _,_*' and expects two integer numbers in the range of 0 to 99. If one of the numbers is out of range, the prompt is repeated.

5 NEW NAME (line 3790) This function prompts: 'Please enter new name (max. 18 characters)' and displays the current entry in the register name string. The string can be edited and saved by pressing (end line).

6 ERASE REGISTER (line 3790) This softkey is used to erase one register. After this softkey has been pressed, the software prompts for a register number.

7 INTERPRETER (line 3800) This softkey is used to activate the command/operation interpreter which is described on page 19. Using the interpreter macros can be edited and run.

8 MORE SOFTKEYS (line 3810) This softkey activates the second set of softkeys.

2nd set of softkeys
The second set of softkeys provides the extended set of workfile edit functions:

1 ADDITION MULTIPL. (line 3720) This softkey is used to perform an addition / multiplication according to the equation in the upper part of the Data Handling menu.

2 SWAP WORKFILE (line 3730) This function is used to back up the active workfile and to read in passive workfiles into the active space. A first prompt shows:
_:________ please enter workfile to be written
and expects the user to fill in a valid drive and filename. A second prompt shows:
_:________ please enter workfile to be read
and expects the user to enter a filename of an existing passive workfile. If both inputs

have been done, the software writes the active workfile to a file with the first filename and then overwrites the active workfile with data from a file with the second filename.

3 SMOOTH (line 3740) With this function excessively noisy data can be processed with three different smooth intensities (1-3). Pressing softkey 3 prompts: 'Please enter register and smooth intensity e.g. (86,3)'

4 EDIT DATA (line 3750) This softkey activates the Edit Data menu which is described in a following subsection.

5 RESET POINTERS (line 3760) This is a development function. It is used to reset all screen update pointers so that all are newly updated. This is useful if the menus do not show the same values as are stored in memory.

6 OPERATION ADJUST (line 3770) With this function the spacing of the X-values of one register is adjusted to the spacing of another register so that it is possible to perform mathemathical operations with these two registers thereafter.

7 STATUS REGISTER (line 3780) The status register function is used to display all integer and real variables stored together with the data curves. It displays a string containing directory entry, date, enzyme, and substrate name.

8 FIRST SOFTKEYS (line 3790) This key changes between the first and the second set of softkeys.

4.3.7 Help Menus

HELP procedure (lines 6000ff)

The HELP procedure is able to show one page out of the electronically stored user manual. When called by the user with the Help Menu touchfield, it shows the first page which contains the table of contents of the help file. A set of softkeys enables the user to page through the document and to exit back to the calling menu. The softkeys are labelled using the KEYLABEL procedure (lines 8700ff). One page of the help screen contains 44 lines. The different pages are read with n times an offset of 44 lines (lines 6515-6570). During the generation of the screen the display remains dark (ALPHAOFF / ALPHA). Lines 6700 - 6782 contain the procedure HELP_COMMANDS which is used to control the help display:

Softkey#1:	former page:	page% = page% - 1
Softkey#3:	next page	
Softkey#4:	goto content	page = 1
Softkey#5:	goto page	prompts *'Please enter page (1-40)'*
Softkey#7:	upper/lower	toggle upper%
Softkey#8:	exit help	show original menu

QUICK_HELP procedure (lines 6900)

This routine is called with the 'IHELP' parameter to decide which help screen appears on the screen. The active menu pointer is saved (isme), the softkeys are labelled: softkey#1 exit softkey #8 more help and the routine is left after the active menu pointer has been set to five.

Each time a softkey is pressed while a help screen is displayed the procedure QUICK_HELP_CMD is activated. There are no touchfields active in this menu.

4.3.8 Edit Data Menu (lines 17000ff)
The EDIT_MENU procedure is called with the source register as a parameter. First it reads data and header of the source register and labels the softkeys (lines 17000 and 17010). The screen is cleared and a header line is printed. Then, on the next line 8 integer header variables are printed. (ivar(i,zr%)). They are followed by 8 real header variables (nvar(nvar(i,zr%)). Next, the column headers for the data points are printed (line 17035 right half). Then up to 64 data points in two columns are printed. The format for the printout as well as the column headers are user definable (system file editor). After the screen has been displayed control is given back to the main program.

The date is changed with the EDIT_CMD procedure. If the Edit Menu is active, this procedure is called after each user input. It needs the TSOFT parameter to decide which subfunction is executed:

1 EXIT EDIT (line 17520)
This function stores the header and data in the workfile, updates the directory and leaves the Edit Data Menu.

2 INPUT X-VALUE (line 17540)
This function prompts: *'Please enter point?'* and expects the number of one data point. The X-value is shown on the second line of the menu where it can be edited. (line 17700). The modified data point is displayed in the menu before the function is left.

3 INPUT Y-VALUE (line 17560)
This function prompts: *'Please enter point ?'* and expects the number of one data point. The Y-value of this data point is shown on the second line of the menu

where it can be edited (line 17700). The modified menu is displayed before the function is left.

4 DELETE POINT (line 17580)

This function prompts: *'Delete which point ?'* and expects the number of one data point. All points following the selected point (I2) are copied one position back (line 17585). Again, the modified menu is displayed before the function is left.

5 INSERT POINT (line 17600)

This function prompts: *'Insert which point ?'* and expects the number of one data point. This number is returned by the procedure (line 3960) in the H1 variable. Next, the data point has to be entered (*'Please enter X,Y?'*) and all points starting with the number I2 are copied one position downwards (line 17610). The new point is copied from variables H1 and I2 into the data array.

6 EDIT TEXT (line 17620)

This function displays all text stored in the haeder file for the active register. The text can be edited and stored.

7 INTEGER VARIABLE (line 17640)

This function prompts: *'Please enter integer variable ?'* and expects a number from 1 to 8. The integer variable can then be edited.

8 NORMAL VARIABLE (line 17660)

This function prompts: *'Please enter normal variable ?'* and expects a number in the range from 1 to 8. The normal variable can then be edited.

4.3.9 Miscellaneous Procedures

GET_DIRECTORY procedure (lines 4200ff)

This procedure reads the internal fast access directory. The routine supports several drives and sorts the files according to file types. The maximal number of entries per filetype on one drive is 128. The Input Menu shows 30 file names at a time. The lower half of the menu can be rolled with increments of 24 files (pointer%(8) line 4320). With five different filetypes 640 files can be accessed per drive.

BACK_OUTPUT procedure (lines 4300ff)

If the OUTPUT_ID_PROCESS pointer is set, (pointer%(5) = 1) and the file has not reached the EOF mark, the next command line is read from the spoolfile and sent to the plotter (4315 print #12, io$). The output is performed with the usual MS-DOS print# function. This function has a few disadvantages:

The operating system MS-DOS 2.11 and the BASIC compiler maintain a I/O buffer with a length of 256 bytes. All commands to the device are buffered in this area and sent to the device when the buffer overflows or the file is closed. Control is not returned to the application until the plotter has worked through all commands. This obstructs the multitasking facility. Response times are improved by adding blanks to short commands (line 4312).

SPOOLER procedure (lines 4350ff)

Commands can be sent to the background spooler by activating the spooler user interface (touchfield or application softkey). When called, this small menu shows the number of plots waiting to be output. Then it accepts a command :

d:	delete active spoolfile
r:	repeat output
c:	clear output queue
p:	pause with output
s:	start output

The spool output from a particular spoolfile is stopped due to several reasons: An EOF mark is read, a delete active command spoolfile, or a pause output is received. To do this, the CLOSE_SPOOLFILE function (line 4400) is called. This function puts the plotter pen back and moves to position 0,0. Both spoolfile and plotter buffers are closed. If no repeat is desired, the current spoolfile is deleted and the queue is moved by one. If a next spoolfile has to be output, a plotter file and the new spoolfile are opened again (line 4415). Then the first command line is read. If this line contains 'STOP' the output is paused and the user has to start the spool again manually.

CHECK_INPUT procedure (lines 5000ff)
In the direct mode the main routine checks keyboard and touchfields if the user is giving an input. This is done in the CHECK_INPUT procedure which starts at line 5000. If the touch sensing mode has not yet been switched on it is switched on, (line 5000). Then during every 1/TSHARE cycle the next command is sent to the plotter (see BACK_OUTPUT procedure). Next the keyboard buffer is read (a1$=inkey$). If it is empty, the command byte is cleared and command codes are set to zero (a$=" ", a=0, b=0). Otherwise, the first byte is converted into a softkey position code (a = asc(a$) - 17) or a touchfield identifier (b = asc(a$) - 89) and the *'command received'* flag is set (IFLAG=1). On line 5020 the command identifiers are cleared. Then the command is checked if it has been a softkey (line 5025) or an application key (lines 5030-5500).
In addition to the softkeys which have a different function in all menus, the MICHFIT software supports APPLICATION SOFTKEYS. These softkeys retain a fixed

function while the application is active. These softkeys are used to select among menus, to switch on the interpreter and to acitvate the help screens. Application softkeys are only active in the direct mode (ilock=0).

On lines 5525 to 5590 the input is checked if it has originated from a touchfield. This routine can be used in two modes: In a first mode the command bytes provided by the TOUCHSCREEN system software identifies the field. Here the TFELD or TARRAY are set. In a second mode the MOUSE OR ALPHA_CURSOR system software provide the position of the cursor when the CURSOR_SENSE function was activated (line 5500). The cursor position is compared with the screen mask. If it falls into a single field or into a field of an array the appropriate pointers are set.

ERROR_RECOVER procedure (lines 6000ff)

Most errors happen while files are accessed: Some errors are due to false file names, others are due to worn out or full discs. The BASIC compiler runs with the /E option provides an ERL variable that contains the line number where the error has occured. This is used to delete file I/O errors and suggest solutions to the user. For errors that happen very rare no solution can be suggested to the user. In these cases e.g. the message: *'Error 21 occured on line 2000. Hit any key to proceed'* is shown. Program execution resumes with the menu that has been active before the command leading to the error has been initiated.

WARN procedure (lines 6300ff)

If certain variables exceed limits, program execution can go on, but it is advisable to inform the user. This is performed by the WARN procedure that is called from the MAIN program each time a command has been executed and returned with the warnz% variable set to a value different from zero.

EMERGENCY_CLEAN (lines 6400ff)

If an application written in compiled BASIC runs out of string or buffer space, it can execute a garbage collection in order to free an area of contiguous memory. This function does not work if sequential files are read in. If not enough memory is available the application inevitably crashes. To prevent such crashes the EMERGENCY_CLEAN routines checks the free string space and removes system file entries from the memory if necessary. This is shown to the application by resetting the SYMBOL_OVERLAY pointer%(25) and the FORMAT_OVERLAY pointer%(26). The system files are then read in the next time they are needed.

STORE_MENU procedure (lines 6800ff)

Before a modified menu can be left, the modified alphanumeric screen has to be written to the screen file. The active menu pointer (IME) is checked if it is in the valid range for an updateable menu. Then, depending on the acitve menu, the storage offset (ISTORE) is calculated. The IPP variable is the pointer to the flag (line 6850ip (ipp)) which decides if screen file and screen are different or not. If the screen does not need to be stored, the routine is left. Otherwise the alpha memory is copied to the I/O buffer with the command STOSCR and the buffer is written to the file put#14, ISTORE.

SMOOTH procedure (lines 7000ff)

The smooth procedure is called with two parameters; a pointer (zr%) to the data in the workfile and an integer (SMOOTH) in the range from 1-3 which selects among three smooth intensities. First the data is read

with the DATA_READ procedure (lines 9200ff) into the array (re()). Then the entries in the array are shifted, so that three points can be generated by extrapolation (line 7020). Similarly, three more points are generated by extrapolation on the other end of the array (lines 7030, 7040). Then, depending on the SMOOTH variable one of three routines is selected.
Routine 1 performs a moving average with three data points (line 7050)
Routine 2 uses five data points to define a parabola.
Routine 3 uses seven data points to define a higher order function.
READ_PLOTFILE procedure (lines 7500ff)
There are three possibilities to access the plotfiles; the first possibility prompts for a file name (line 7500) and for the first destination register in the workfile. It copies the file name into the FILENAME$(1) string array and the first destination register into the TQUE(1) array. The name of this procedure is INREAD_PF. It then calls the READ_PLOTFILE procedure (line 7520) with the common parameters FILENAME$(1) and TQUE(1). This routine tests if the file exists and opens it. The actual read is performed by another subroutine (READ_PF1) which starts at line 7522.
First the EMERGENCY_CLEAN routine is called to ensure an area of 1000 bytes of continous memory. Then an integer (DFORM) is read from the file. This integer identifies the program and version which has generated the file. If the new version is compatible, the plotsize variables are read (lines 7530-7552 or lines 7740-7770). Two different reading routines have been included into this procedure so that even older files generated by earlier program versions can be read. The version starting with line 7740 is the latest version,

for this reason only this version is described here. (See WRITE_PLOTFILE procedure for information on plotsize variables and date structure of plotfile). Reading is carried on with line 7780. Here, the output transformation and the macro flag are read. On the same line the display is selected as the output device (TAUS=1) the number of data curves is copied from the array pg(8) to the pointer ANZMT. Automatic scaling is switched off and the default flag is set. If the user definable output transformation is selected (TMOD=6), the input string for the transformation interpreter is read (line 7790). If the macro mode is on, the macro memory together with variables and strings is read. Next from 0 to 10 datafiles are read from the plotfile and stored into the workfile (lines 7800-7850). The header information is read on line 7810 and stored to the workfile on line 7815 with the command put#8, zr%. Then, additional output information is read (pen, symbol/linetype, and interpolate on/off). Next, the data is read from the plot file and written to the workfile with the procedure HEADER_WRITE. The routine is left when the file has been closed and the update pointers have been reset.

The procedure UPPERCASE converts lowercase letters in the string in $ into uppercase letters.
WRITE_PLOTFILE procedure (lines 8000ff)
The WRITE_PLOTFILE routine prompts: *'Please enter file name (max. 8 characters) ?'* and inputs 8 characters using the GETSTRING procedure (line 40'000ff). The string is converted to uppercase and checked if it is longer than 2 characters and shorter than 9 characters. File names with RESCUE, EXIT, or PLOT are neither accepted (lines 8000 to 8010). The new file name is

compared with existing file names of the same file type. If the file already exists, the function RESCUE is called (line 7950). This function renames the old file to a file with RESCUE. and the plotfile extension. This procedure prevents loss of the file until the next file is overwritten. If a new file is written the name is added to the directory and the modified directory is read again (line 8025). Then the file is opened and the procedure WRITE_PF1 is called.

WRITE_PF1 procedure (lines 8035ff)

The WRITE_PF1 procedure is either called by the WRITE_PLOTFILE procedure or by the CURR_STAT_SAVE procedure. First all plot strings (BESCHR$()) are checked if they contain <End line> characters (lines 8035-8050). Carriage return characters are replaced by spaces to avoid crashes of the BASIC software. Then the data format code (DFORM) is written. This is followed by a block of five axis definitions pxu#(1-5), pyl#(1-5), pxo#(1-5), and pyr#1. The first entry in the array defines the start of the axis, the second entry defines the stop of the axis, the third is the increment between labels resp. ticks and the fourth entry is the distance between the start of the axis and first label. The fifth entry is the axis control variable. If it is set to zero, no axis is plotted. If it is set to one, just the ticks are plotted. If it is set to two, just the digits are plotted. If it is set to four, only the label of the axis is plotted. All parts of the axis are plotted, when the axis control variable is set to seven.

The next block is composed of 5 arrays with 30 entries each. The pg() array contains plotsizes (X-size, Y-size, left rim, lower rim), speed of plot pen, tick size line spacing, and the direction of the annotate

labels. The pgx() and the pgy() arrays contain the sizes of the characters, the position of the digits, labels, and annotate lines. The tl() array contains the length of the strings in bytes. The pga() array contains the width of the columns.
The third block contains the data for the draw procedure. The length of this array is determined by the IDMAX variable which is printed just before the start of this array. If IDMAX is zero no array is written. The fourth block contains all plot labels (BESCHR$(1) - BESCHR$(20)). Then the type of the output transformation is stored.
In the macro mode (comfile%=1) the current macro is saved to the plotfile too (line 8140). In the same way the strings (p$()) and the constants (c()) are stored. On line 8150 the workfile is accessed and on line 8155 the header information is converted into ASCII and added to the sequential file. Then the symbol, the pen, and the interpolate status of the curve are saved. On line 8165 the data is stored in the form of an X Y list.
The loop from 8150 to 8190 is executed up to 10 times before the plotfile is closed and the procedure ends.

ADD_MUL procedure (lines 8200ff)
The header information is copied from source register A to destination register C (lines 8200, 8205). If the number of data points in both source registers is not identical the OPERATION_ADJUST procedure is called (line 8206). The header strings are copied to the destination register and read into the common data area.
Then the data of source registers A and B are read into the arrays dy(1,) / dy(2,). The operation is

performed within the loop on line 8240 - 8260. The result is stored with the DATA_WRITE procedure. The Y-max. variable is modified (nvar(3,)) and stored with the HEADER_WRITE procedure.

KEYLABEL procedure (lines 8700ff)
The content of the string h1$ is sent to the softkey label fields by means of the escape sequence: ESC &foa1K16d1L. 16 characters containing the label are added to the string and printed to the terminal for each softkey. Labels are shown on the display with the ESCAPE sequence ESC &jB.
SET_TOUCH procedure (lines 8500ff)
The information for the touchfields is taken from the systemfile and written into the ICON () common integer array. The integers are composed of the COL * 100 + ROW. A single field is defined by the upper left and the lower right corner. Up to 25 single fields can be defined. The response character of single fields ranges from A to Z. The touchfield is set up with the FNTOUCH function of the system software.
Field arrays are defined by one integer (ANZZ * 100 + ANZS). The shape of the individual fields is given as for the single fields. The array is defined when the first row of fields and the first column of fields has been read.
READ_DEFAULT procedure (lines 1800ff)
With the READ_DEFAULT procedure one plot setup can be read from the systemfile for each transformation. This procedure reads the labels (line 18060), the user units of X and Y axes (lines 18070 and 18080) from the systemfile. It is called from the Output Menu after the output transformation has been changed by touching the DEFAULT single field. An alternative to read in plot

default is reading a user defined plotfile with no data.

4.3.10 Workfile Input/Output

The workfile access is done with four procedures:

DATA_WRITE procedure (lines 9100ff)

```
if zr%>100 of zr%<1 then zr%=1      the size of the
                                    workfile is limited
                                    to 100 registers
h4$=space$(253)                     the I/O buffer is
                                    prepared
for wl=1 to 63                      the data is
                                    converted into
                                    binary
    mid$(h4$,wl*4-3,4)=mks$(re(wl)) format and
                                    assembled in h4$
next wl                             (first the Y
                                    values)
lset e1$=h4$                          and copied to the
                                    I/O buffer
for wl=101 to 163                   similarly the X
                                    values are
                                    converted:
    mid$(h4$,(wl-100)*4-3,4)=mks$(re(wl))
next wl
lset e2$=h4$                         and written to the
                                    I/O buffer
put#9,zr%                           This command copies
                                    the buffer to
return                              to the disk and the
                                    procedure ends
                                    DATA FILE:
             <username>.KIW
```

```
DATA_READ procedure (lines 9200ff)
if zr%>100 of zr%<1 then zr%=1        Only registers from
                                      the initialized
                                      part of the file
                                      can be read
get#9,zr%                             random access to
                                      file
for wl=1 to 320                       Clear data array
   re(wl)=0
next wl
for wl=1 to 63                        Convert binary
                                      information in
                                      buffer
   re(wl)=cvs(mid$(e1$,wl*4-3,4)      to floating point
                                      variables
next wl                               (first Y - data)
for wl=101 to 163                     Convert binary
                                      information in
                                      buffer
   re(wl)=cvs(mid$(e2$,(wl-100)*4-3,4)
next wl                               (second X - data)
return                                end of procedure

HEADER_READ procedure (lines 9800ff)
if zr%>100 of zr%<1 then zr%=1        Only registers from
                                      the initialized
                                      part of the file can
                                      be read
get#8,zr%                             random access to
                                      file
for wl=1 to 10                        The binary
                                      information in the
                                      buffer is
  ivar(wl,zr%)=cvi(mid$(vat$,wl*2-1,2))
  nvar(wl.zr%)=cvs(mid$(vat$(wl*4+17,4))
next wl                                  converted into the
                                        10 integer and
                                        normal variables
return
HEADER_WRITE procedure (lines 9900ff)
if zr%>100 of zr%<1 then zr%=1          the size of the
                                        workfile is limited
                                        to 100 registers
get#8,zr%                               random access to
                                        file
hi$=mid$(vat$(61,70)+"               " The comments are
                                        preserved
h4$=space$(130)                         a buffer is prepared
                                        to assemble the
for wl=1 to 10                          binary data
    mid(h4$,wl*2-1,2)=mki$(ivar(wl,zr%))
    mid$(h4$,wl*4+17,4)=mks$(nvar(wl,zr%))
next wl
mid$(h4$,61,70)=mid$(h1$,1,70)          The comments are
                                        added to the new
lset vat$=h4$                           buffer and the
                                        entire information
put#8,zr%                               is copied to the
                                        disk
return                                  HEADER FILE :
                                        <username>.KVA
```

SET_WFUPDATE procedure (lines 9950ff)
If one of the workfiles is changed by any routine, a pointer to this workfile is set. This is used in the Data Handling and the Output Menu to update the fields if necessary.

4.4 Input Overlay

The input overlay implements the function which are needed to read from the different data sources and to accept the different filetypes. These functions are called from the Input Menu. There is one exception the SWAP_WORKFILE procedure is called by the Data Handling Menu. This procedure copies the active workfile onto a file and reads in stored workfiles into the active space.

When control is given to the input overlay a series of local variables is defined (lines 222 - 227) in addition to the common variables defined in the included file 'COMVAR'. A few local variables are initialized and the workfile buffers are declared (lines 230 and 370). Furthermore, the I/O buffer #11 is opened for the line printer. The variable FORTS decides if the SWAP_WORKFILE procedure (FORTS = 3) or the other input procedures have to be executed (FORTS # 3). Lines 1500ff select among different sources. If the source variable TEIN is set to 4 the keyboard input routine is activated. Otherwise file input routines are used. Line 1600 selects among different file tpyes:

1 = X - Y data file
2 = -
3 = spectral data
4 = kinetic data
5 = (plot files) not in this overlay

4.4.1 Keyboard Input

KEYBOARD_INPUT procedure (lines 1900ff)

The KEYBOARD_INPUT procedure accepts header information and data so that workfile registers can be entered manually.

Name of register	1 - 18
Series and number e.g. A1	19 - 20

Date	21 - 30
Enzyme name	31 - 50
Enzyme concentration	nvar (1,)
Substrate name	51 - 59 + K
Substrate concentration	nvar (2,)
Measure range (2 entries)	ivar (1,) ivar (2,)
Normalizing range (2 entries)	ivar (7,) ivar (8,)
Number of data points	ivar (5,)

From the entered text the header string is assembled.
Name of register.AlDate .Name of enzyme Name of sustrate ...K

The enzyme concentration and the substrate concentration are stored in the real variables nvar (1,) and nvar (2,). The measure range is stored in the integer variables 1 and 2 and the normalizing range is stored in the integer variables 7 and 8. The strings are entered via the INPUT_NUMBER procedure (line 3959) (see VKI listing). On lines 1925 and 1950 default values are stored in the rest of the header variables: Output window: ivar (3,) = 1, ivar (4,) = ivar (5,) + 1, Y-max = nvar (3,) = last data point. Then the register is stored and the KEYBOARD_INPUT procedure ends.

4.4.2 Read and Average from Files

READ_AVERAGE procedure (lines 10000ff)

After having cleared the display, this procedure reads the header of all serial input data files (line 10010). It then displays the header information so that it can be modified by a user (line 10030). If the user has not pressed 'a' for abort (IENDE = 0) the header variables are copied to the header of the workfile register (line 10050). The data is read (line 10070). The data is stored in the workfile and the maximal Y-value is calculated as the average of the last three data points (lines 10300 - 10320). The header information is stored and the procedure ends.

HEADER_IN procedure (lines 11000ff)
This procedure reads header information from the input file in ASCII and converts it and stores the variable in the array var (,) (line 11040).

HEADER_CHECK procedure (lines 11300ff)
This procedure displays the header information of up to 5 input files (lines 11300 - 11317). Functions are provided to change header variables (line 11430 - 11480).

COPY_HEADER procedure (lines 11500ff)
This procedure copies the header information of the first input file into the header of the workfile register.

IN_SERIAL procedure (lines 12000ff)
This procedure reads the serial files (line 12070) and adds up to the Y1, Y2 and X data into the array re( ). The data is read in a floating point ASCII format with +5.8756E+001 four digits in the functional part and a three digit exponent. The exponent is then shortened to two digits and the number is converted by the val(x$) function of the compiled BASIC language (lines 12080 - 12090). The files are closed and the average is calculated (line 12110). Next the data is stored in two registers (range 1 and range 2).

READ_XYDATA procedure (lines 19000ff)
This function is used to import data that has been stored by other applications. It needs a series of file

names to be passed as parameters. On lines 15010 it decides if several data curves have to be read from the same file or if a new file has to be opened. One or serveral lines of header information (starting with /// or ***) are read. The header information has to be followed by an integer number determing the number of data points that have to be read following this number (line 15054). The number of data points has to be in the range from 1 to 500 (line 15055). If this is the case they are read using the loop on line 15060. The data is accepted in two formats:
In a floating point exponential format
+7.1438E+001+2.3798E+000 or
in a printer format 275.5 = 7.0821E+01.
In both cases the X-value preceedes the Y-value. If it is necessary any other data format can be included into this procedure.

4.4.3 Swap Workfile
SWAP_WORKFILE procedure (lines 3000ff)
This procedure prompts for two file names.
a) the workfile to be activated and
b) the file name where the active workfile shall be stored

A few procedures are already familiar from the Data Handling Overlay :

ASCII_IN procedure	40'000	see VKI
HEADER_READ procedure	9'800	see VKI
HEADER_WRITE procedure	9'900	see VKI
DATA_READ procedure	9'200	see VKI
DATA_WRITE procedure	9'100	see VKI
COMMAND procedure	4'000	see VSP
KEY_IN procedure	5'000	see VKI

IN_NUMBER procedure	3'950	see VKI
BACK_OUT procedure	4'300	see VKI
CLOSE_SPOOL procedure	4'400	see VKI

If file names have been entered properly, the header of the workfile (lines 3220 - 3260) and the data of the workfile (lines 3320 - 3360) are copied onto the new file. On line 3500 the second file name is used to open a passive workfile and to copy it into the active zone. First the header information is copied and checked if it contains valid workfile data (lines 3530 - 3560). After 99 entries have been copied, the header file is closed and the data is copied (lines 3630 - 3660). Then the header information is read from the new workfile into the common data area. After that the procedure ends.

4.5 Fit Overlay RKI.BAS / RKI.EXE

When the fit overlay receives control, it defines several local variables:

4.5.1 Local variables

re (400)	The re() buffer holds the measured data during fits. This buffer is used for several other purposes during the time this overlay is active.
ireg (11,11)	This two dimensional array stores the fit model, the source register, the fit limits,
erww (11,11)	This two dimensional arrays stores the first estimates of all parameters, for a fit in the first

	dimension. The second dimension is used to hold several fits in a buffer.
raster (11,11)	This two dimensional array stores the finest grid for all parameters. The grid is used to define a termination criterion for the fit as well as the grid for the first fit.
fak% (11,11)	The fak% two dimensional array stores the expansion factor for all parameters of a fit. The factor is used to expand the finest grid (raster) for the first grid. If all factors are 1 the fit has converged.
rex()	Multi purposed buffer
rey()	Multi purposed buffer
zr(10)	This array stores the register numbers of the data curves. Up to ten curves are queued in a job queue.
ik(10)	
ist(10)	
interpk(10)	Interpolate status of data curves
lbeg(11)	These variables hold the start, the
lend(11)	stop and the interval of the currently
lstep(11)	active grid - or gradient

	search.
fit% (11)	Fit control variable
pne% (11)	Positive/negative flag
da% (11)	Positive/negative flag
drucker%=1	Here the printer level
b$="e"	Input byte during fit
offset%=0	Roll status of model list
running%=1	Fit overlay is running
iplot=0	
ilock=0	Main Menu is enabled
pointer%(26)=0	Main Menu can be used
exp$="+####^^^^"	Exponential format
idmax=pointer%(2)	Draw pointer comfile and
comfile%=pointer%(3)	Interpreter flags are copied
interpreter=pointer%(4)	Into local variables
dbez$ "a:b:c:"	Idrive pointer can select among drive a: drive b: or drive c:

Lines 300 to 342 initialize strings in the common data area. They are used to show the Fit Input Menu later on. This approach has been selected to facilitate a possible read of these strings from the systemfile. No sytemfile for the fit overlay has been written, however. In a similar way the strings REMOD$ and REMOVE$ are used for printouts later in the program. Lines 350 to 355 define the string for the short names of the parameters. The first column is for the first parameter of all models. Lines 360 to 370 define format strings that are used during printing. Line 400 prevents that the program can be executed by calling from the operating system. It can only be activated if the PINIT variable has been set by the VKI or any other program. On line 410 the workfile I/O buffers are defined. On line 420 a printer I/O buffer is opened.

4.5.2 Fit Menu

The main program has a similar structure as the main program in the VKI program. It defines an infinite while - wend loop that can be terminated by resetting the RUNNING% variable. If this happens, all strings that are only used by this overlay are erased, a garbage collection is initiated (line 820), and the calling program is reactivated (line 890).

The directory line routine is the same routine as can be found on lines 3300ff in the Data Handling overlay. The fit overlay uses another set of softkeys. Softkey functions are implemented in the FIT_CMD procedure.

FIT_COMMAND procedure (lines 3400ff)

This routine accepts only softkeys as inputs (TSOFT has to be in the range from 1 to 8). The following paragraphs describe the actions that are taken when softkey 1 to 8 are pressed in the Fit Menu:

1 SIMULATION (line 3520)

A simulation is performed on a user definable data curve. If no additional input is given, a curve is generated using the fit type and the fit parameters stored in the header of the data curve. A simulation can be generated using the X-points of the data curve and entering models and parameters at will (see SIMULATION procedure on line 5000).

2 INPUT SINGLE (line 3540)

This softkey is used to enter the first estimates, grids, and factors for one fit. To do this the input has to be placed in one to the 10 fit buffers. In response to the prompt: *'Which position in batch job (1-10)'* a number in the range from 1 to 10 has to be entered. This number is used to store the parameter values in the appropriate fields of the two dimensional array.

3   INPUT MULTIPLE (line 3560)

This function clears the fit bufffer (lines 3560 - 3564). Then a FOR - NEXT loop is used to enter up to 10 fits into the fit buffer. To do this the SHOW_MODEL and the INPUT_FIT procedures are called within this loop. After the input has been finished (after 10 inputs or if a model 0 is selected), the fits have to be started by pressing softkey 4.

4   EXECUTE _FIT (line 3580)

This function executes the fits compiled in the fit buffer until 10 fits have been performed or a model 0 is found. The fit is executed by the FIT procedure on line 5200.

5   LINEAR_REGRESSION (line 3600)

This softkey starts a linear regression. To do this the procedure LIN_REG is called. This procedure performs fits to X-Y data or transforms data before fits (e.g. Eadie Hofstee transformation, double reciprocal transformation).

6   ROLL_MODELS (line 3620)

This softkey is used to switch among the upper half of the model display (models 1-9) or the lower half of the model display (models 11-19).

7   ENTER EQUATION (line 3640)

This softkey is used to change the equation of the user definable model (model 10). Line 3640 clears the command line, line 3645 shows the prompt message, and line 3647 is used to modify the current equation stored in FORMAT$(20). The equation is converted to uppercase and stored in FORMAT$(20). Line 3395 clears the command line.

8 EXIT (line 3660)
This function exits from the Fit Menu to the Output Menu (FORTS = 2). Setting RUNNING% to zero stops the main loop. Setting PINIT to 1 allows admittance to other overlays.
4.5.3 Fit Procedures
RECOVER procedure (lines 4200ff)
This procedure is used to show error codes, error lines, and error messages to the user. Apart from the error messages this procedure is identical to the procedure in the VKI overlay.
MODELS procedure (lines 4900ff)
This procedure displays a small menu with fit models on four lines below the directory lines of the Fit Menu. The menu window changes depending on the variable OFFSET which can be changed by softkey 6.
INPUT_FIT procedure (lines 5000ff)
The first input consists of four entries:
IREG(1,) the fit type 1 - 19
IREG(2,) the source register containing the experimental data
IREG(*,) the first data point to be used for the fit
IREG(4,) the last data point to be used for the fit
The input is checked on line 5070 if the fit model is in the range from 1 to 19, if the source register is a valid workfile register (1-99), if the first data point is 1 or higher, and if the first and last data point are within the range of stored experimental data points. The variables are shown again on the screen (line 5105). Next the destination register for the simulation (IREG(6,)) amd the maximal number of fit cycles (IREG(7,)) has to be given. Then the register containing noise (IREG(5,)) has to be given. The noise is used to calculate an individual weight for each data

point. If 0 is selected, each data point is equally weighted. Otherwise, the noise register is checked if it contains the same number of data points like the register containing the experimental data (line 5130). The FOR - NEXT loop on lines 5150 to 5165 is used to enter the first estimates, the finest grid (max. accuracy), and the expansion factor for up to 6 parameters. The content of the string with the short names of the parameters (line 350) is used to decide if an input is necessary (line 5150). Line 5155 generates a prompt string and adds the entry of the first estimate resp. result variable (NVAR 4+j, zr%). The INPUT_3NUM procedure (line 7100) is used to enter the numbers or assign default values. The INPUT_3NUM procedure returns the first estimate in H1, the finest grid in H2 and the factor in H3. Line 5160 accepts only a binary number as an input for the factor. If it is different, the input has to be repeated. On line 5165 the fit type variable of the selected register is set. After having cleared the command line the INPUT_FIT procedure ends.

FIT procedure (lines 5200ff)

When the FIT procedure is activated, the user has to decide with which position of the batch the fit should start (1-10). The position (ii) defines the start of the FOR-NEXT loop on lines 5200 to 5450. The fit type (REGART), the noise register (STOEVREG), the first data point (VON%), and the last data point (BIS%) are loaded from the batch buffer. If the noise register is zero, the weight buffer dy(1,) is loaded with n times 1 (line 5232). Otherwise the noise information is read into the buffer dy(1,). Then the experimental data is read into re( ). The procedure WEIGHT (line 7500) converts the noise into individual weights and the

procedure FIT_DEBUG shows the menu mask of the screen, which is shown during fits (line 12000). On line 5240 the finest grid variable RASTER (j,ji) is checked if one fo these variables is zero the fit is skipped. In the same loop the fit control variable FIT%() are set to initial values (99).
Line 5350 is the first line of the infinite FIT loop. The last line is 5350. This line provides a means to change the fit equation while the fit procedure is running (line 3645 inputs the equation after the user has pressed the 'f' key). In a similar way the interpreter can be activated by pressing 'i'.
The next line calls the procedure SET_LIMIT (line 5600) to set the lower and upper limits for the active search cycle. Then the loop counter is incremented and the stop conditions are checked. If a stop condition is met, the loop is left after the message *'Fit terminated early'* has been printed on the line printer. From line 5290 the FITC procedure (line 6700) is called. This routine returns the best parameters in the common variables NVAR (5,) to NVAR (10,). They are copied into the ERWW( ) array if no user input of the estimate have been given (ERWWFLAG% = 0). The routine CHECKV is called to check the estimates and fit control variables. Then using the routine EVALUATE, the result of the fit is analyzed. This routine checks if the fit has converged (FLAG% = 1) or prepares the variables for the next cycle. If the fit has converged the PRINT_FIT procedure is called (if the DRUCKER% variable is set). Next the SIMULATION2 procedure is called. After that the next curve is being fitted (end of FOR - NEXT loop). If the last fit has been performed, the command line is cleared, the softkey labels are changed and the FIT procedure is left.

CHECKV procedure (lines 5500ff)
This procedure checks if the estimates are zero or exceeds (9999). The FAK% variables are checked if they exceed 4096. In case of an overflow an error message is printed to the invalid result and the next fit is started.
SET_LIMIT procedure (lines 5600ff)
This procedure consist of a FOR - NEXT loop (from 5600 to line 5730), that sets the fit limits LBEG( ) LEND( ) and LSTEP( ) for all parameters. Lines 5600 and 5610 handle special cases where one parameter is set to 1 (FAK%( )=-1) or kept at the estimate (FAK%=0). Line 5640 handles the starting condition (FIT%( )=99). Line 5650 decreases the amount of the fit control variable (the tendency of the fit). Lines 5655 to 5675 define a different search grid for each setting of the fit control variable. If the fit control variable exceeds +3 or -3 (three times the same tendency in a sequence) the FAK%() variable for that parameter is multiplied by two to increase the speed of the gradient search. Then the LSTEP() variable is calculated as the product of the RASTER() and the FAK% variable. The LSTEP variable with the I2 and I3 variables (which have been derived from the tendency variables FIT%()) are used to calculate LBEG() and LEND(). Line 5725 prevents LBEG from changing the sign in most kinetic fits, this would produce invalid results. After the limits for all parameters have been set the FIT_DEBUG screen is updated and the SET_LIMIT procedure ends.
EVALUATE procedure (lines 5800ff)
The FOR - NEXT loop from line 5810 to line 5920 compares the results of the fit (ERRW()) with the upper and lower limits. If the estimate is identical with the

lower limit, the FIT%() variable is decreased by 2. If the estimate is identical with the upper limit, the FIT%() variable is increased by 2. In both cases the number of limited estimates is increased (ANZAN%). Line 5850 prevents estimates from changing the sign. Line 5860 detects the highest factor of all parameters (MAXFAK%) and if any tendency variable is different from zero (H1). After all parameters have been analyzed, the MAXFAK% and the H1 variables are used to decide if the fit has converged. In this case a FLAG% variable is set. Before this procedure is left, the FIT_DEBUG Menu is updated.

SIMULATION procedure (lines 6000ff)

The simulation process is divided into two procedures: The SIMULATION procedure inputs the necessary parameter, the SIMULATION2 procedure executes the simulations after a user input or after a call from the FIT routine. The user has to select one of the equations (1-19) and one of the registers (1-99), after the Fit Menu has been shown on the screen. All parameters have to be entered (lines 6040 - 6046). The parameters are stored and the experimental data are read. Then the destination register for the fit has to be selected.

SIMULATION2 procedure (lines 6150ff)

All header information is copied from the source register to the destination register. The header variables are checked if the register can be used for a fit. If the experimental curve contains less than 21 data points, more points are generated for the fitted curve by interpolation (EXPAND procedure). Next the theoretical curve is simulated with one of the equations in the CALCULATE procedure. Line 6250 generates the directory entry for the simulated curve and stores header information and data. The user definable equation (REGART = 10) causes the interpreter to be called (line 6215).

CALCULATE procedure (lines 6500ff)

This procedure contains all the equation for the fitting procedure. When called the parameters p1 thru p6 have to be passed together with the substrate concentration S. It returns the result in the SIGNAL variable.

Model	Name	Equation
1	Michaelis Menten	SIGNAL = (p1(1+p2/S))
2	Consecutive Reaction	SIGNAL = (p1*S+p2*S^2)/(1+p3*S+p4*S^2)
3	2 Km 2Vmax values	SIGNAL = p1*S/(p2+S)+p3*S/(p4+S)
4	Hill equation	SIGNAL = (p1*S^p3/(p2^p3+S^p3))
5	Noncooperative sites	SIGNAL = p1*((S/p2+S^2/p2)/(1+2*S/ p2+S^2/p2))
6	Sequential interact.	SIGNAL = p1*((S/p2+S^2/(p3*p2))/(1+2*S / p2+S^2/(p3*p2)))
7	Inhibition	SIGNAL = p1-p3*S/(p2+S)
8	Uncomp. Inhibition	SIGNAL = p1*S/(p2+S*(1+p4/p3))
9	Noncomp. INHIBITION	SIGNAL = p1*S/(p2*(1+p4/p3)+S* (1+p4/p3))
10	User defined	see Interpreter
11	Dependent site mech.	SIGNAL = ((p1+p2*p4*S)*S/(1+p2*S+p2*p3 *S^2)

12	Indep. site mech.	SIGNAL = (p1+p4+(p2*p4+p3+p1)*S)*S/(1+(p2+p3)*S+p2*p3*S^2)
13	Dead end complex	SIGNAL = (p1+p2*p4*S)*S/(1+p2*S+p2*p4*S^2)
14	Exchange mech.	SIGNAL = p1+p1*p3*p4*S/(1+(p2+p3)*S+p2*p3*S^2)
15	Variable inhibitor	SIGNAL = p1-((p1-p3)/(1+S/p2))
16	Linear regressions	SIGNAL = p1+p2*S
17	Double reciprocal	SIGNAL = Michaelis Menten
18	Eadie Hofstee	SIGNAL = Michaelis Menten
19	Hanes Wolfe	SIGNAL = Michaelis Menten

FITC procedure (lines 6700ff)
First, the deviation parameter is set to a high value, and the loop counter is reset. The enzyme concentration is read from the header. The variables TSTERN and TCYCLE are reset so that they can be used to show a bar (of stars) that visualizes the progress of the fit. For that purpose the total number of cycles (TOTAL of loops 1 to 4) has to be calculated (lines 6702 and 6703). Then, the 6 nested loops for the 6 parameters begin. Each parameter covers the range from LBEG to LEND with increments LSTEP. The progress bar is updated after the fourth FOR-NEXT loop. First the counter TCYCLE is incremented and the relative progress is compared with the number of start already on the display (TSTERN). If the new value is higher, the additional stars are printed and the TSTERN variable is set. Line 6723 checks if a user input is pending and executes the command (subroutine on line 7800). After the next loops

the square of the deviation is added either by the FIT_INT procedure or by the loop on lines 6750 to 6760. The parameters leading to the lowest root of squares are stored before the loops are closed.

PRINT_RESULT procedure (lines 6900ff)

This routine uses read in strings to assemble the printout. If the user defined equation is selected (REGART = 10) the equation is printed (line 6935). In case the printer is not switched on, an error message is shown (ON ERROR GOTO 7750).

CHANGE_PAR procedure (lines 7000ff)

This routine is used to temporarily suspend the fitting procedure and to change one or several factors, estimates, grid sizes, or fit control variables.

INPUT_3NUM procedure (lines 7100ff)

A string is input by means of the ASCII procedure. Depending on the number of commas from one to three strings are converted to real variables. (H1, H2, and H3). If no commas are found the variables are determined automatically (lines 7152 - 7158). H1 = estimate, H2 (finest grid) is approximately estimate / 1000, H3 (factor) = 128. If no valid number has been given after the first comma, the parameter is kept constant (H3 = 0). The same applies to lines 7170 and 7175.

EXPAND procedure (lines (7400ff)

This procedure takes the X-values of an experimental data curve and inserts two more data points between existing data points. The modified number of data points is stored in the header variable (INVAR (5,) = w-2).

WEIGHT procedure (lines 7500ff)

This loop on line 7550 and 7560 calculates the average signal to noise ratio (H1). The second loop calculates the individual weight by dividing the signal to noise ratio by the average signal to noise ratio of the selected range ((IVON - IBIS).

TIMEDATE procedure (lines 7600ff)
This routine reads time and date from the internal clock.

PRINTER_OFF procedure (lines 7700ff)
This routine resets the printerlevel pointer (DRUCKER%) to zero, to avoid printing when the printer does not respond. A message is displayed to the user which can select 'r' to retry printing.

FIT_CMD procedure (lines 7800ff)
This procedure executes the softkeys which are displayed in the FIT_DEBUG Menu:
Softkey 1 selects a different print level (0-2).
Softkeys 3-6 call the CHANGE_PAR procedure to modify one of the fit parameters. Softkey 3 changes the factor, softkey 4 changes the estimate, softkey 5 changes the grid and softkey 6 changes the fit control variables. Softkey 7 changes to the next fit after the current cycle has been finished. The message *'Fit terminated early'* is added to the printout. Softkey 8 stops the fitting immediately.
Pressing the 'i' key activates the interpreter and pressing the 'f' key displays the active fit equation so that it can be modified.

4.5.4 Linear Regressio

LIN_REG procedure (lines 10000ff)
First, the command line is cleared and the number of parameter pointer (IPARM) is set to two (see PRINT_RESULT procedure). Then the number of data points and a transformation has to be entered. If no transformation has been selected, the experimental data

are read, the linear regression variables are initialized and the linear regression is performed. From the first to the last point the following variables are summed up (line 10050):

Summ of X-values	SUX
Summ of Y-values	SUY
Summ of X * Y	SUXY
Summ of Y square	SUYY
Summ of X square	SUXX

These summs are used to determine the intercept, the slope and the correlation (lines 10060 - 10080):

rhoY	$= \text{sum } XY - (\text{sum } Y / N)^2$
rhoX	$= \text{sum } X^2 - (\text{sum } X / N)^2$
slope	$= (\text{sum } XY - (\text{sum } X / N) * (\text{sum } y / N)) * \text{rhoX}$
intercept	$= \text{sum } Y - \text{slope} * (\text{sum } X / N)$
correlation	$= (\text{slope} * (\text{rhoX})^{-2}) / (\text{rhoY})^{-2}$

If either the number of data points, 3X or 3Y is zero the message: *'Invalid linear regression'* is displayed and no result is calculated. Otherwise the result of the regression is printed and stored in the header of the experimental data (line 10160). Next, a theoretical curve is simulated using the SIMULATION2 procedure. To do this, a destination register for the simulation has to be entered. Two parameters, the intercept and slope are used by the simulation to generate the line according to the following equation:

SIGNAL = P1 +P2 * S

This is the equation number 16 (REGART = 16) on line 6680 of the CALCULATE procedure. A line would be sufficiently determined by two data points. But if this line is transformed into another mathematical space, it might become nonlinear. For this purpose the line, like any other simulation is stored as an X-Y array of data points in an individual register of the workfile. Lines 10500 to 10950 contain a linear regression routine that sums up data points after they have been transformed. The transformations are:

Name	Model#	Transformation	
Double reciprocal	17	XWERT=1/S	YWERT=1/V
Eadie Hofstee	18	XWERT=V	YWERT=V/S
Hanes Wolfe	19	XWERT=S	YWERT=S/V

Then the linear regression is performed (see above) and slope, intercept and correlation are calculated. Depending on the transformations the Michaelis constant and the maximal velocity are calculated as follows:

Double reciprocal	Km	= slope/intercept
	Vmax	= 1/intercept
Eadie Hofstee	Km	= 1/slope
	Vmax	= intercept/slope
Hanes Wolfe	Km	= intercept/slope
	Vmax	= 1/slope

The Michaelis constant, the maximal velocity and the correlation are stored in the header of the experimental data and printed with the PRINT_RESULT procedure. Again, a destination register for the simulated curve has to be selected before the SIMULATION2 procedure is called. For all simulations the MICHAELIS MENTEN model is used (REGART = 1).

4.5.5 Influence Fit Menu

KEY_INPUT procedure (lines 11000ff)

The KEY_INPUT procedure is a short form of the CHECK_INPUT procedure (see line 5000 in the VKI overlay.

FIT_DEBUG procedure (lines 12000ff)
This procedure is used to display one screen of information about the fitting procedure which is currently active. When it is called, the header information from the workfile is read and used to display the following lines on the screen.

```
**************** DIRECT CURVE FIT ******************

Curve fit to register:  1 Thr 102 (wildtype) 26.Mar. 88
Cytochrome c oxidase : 10.60 nM  Mod. Cyt c
                               Max. conc.:  10.00 uM
Data points used form    :   1  to   20  Register with
                                    abs. noise      30
Regression Type          :   2 Km 2 Vmax values
Batch position number    :   2


The next fit is expected at   Printlevel    1  *******

        Param.1 Param.2 Param.3 Param.4 Param.5 Param.6
Meaning  Vmax     Km
Boundary from     0       0       0       0       0       0
Boundary to  9.9E99 9.9E99 9.9E99  9.9E99 9.9E99 9.9E99
Fit control    +1     +2     +99      +99    +99      +99
Gridfactor    8    16       0       0        0        0
Range from          34.200      0.1600
Range to            35.800      0.1900
Result              35.000      0.1750
Root mean square    1.27895
```

The four lines starting with line 2 of the screen display the header information in the same way as it is printed to the result. Next, the batch position number

of the fit currently in process is shown. On the same line the print level shows how much information will be printed. After a space which is later taken by the bar of stars, the headings of the parameter is shown. For each parameter a short label is added. This label is selected differently for each parameter and each fit model (line 12259). The next lines are provided with the row headings (see below). Before the procedure stops, the softkey labels are changed.
FIT_UPDATE procedure (lines 12500ff)
This procedure is called each time one fit cycle has been finished. First, it calculates the expected duration of the fit cycle and adds it to the time. The variable I1 contains the amount of time in seconds and is calculated as follows: I1 = cycles * data points / 150. If the interpreter is used, this time has to be multiplied by a factor of 8 or by 16 if long equations are used (lines 12540 and 12545). The performance of the interpreter can be increased dramatically if it is used once in a compiler mode. This means the codes generated during an interpreter run have to be stored in an array and used by an small runtime routine. Here the example of the compiled code for a simple equation:
R = (C1*S) / (C2+S)

Code	operand1	operand2	Memnonic
5	1		LOADX,C1
13	-		LOADY,S
103	1		Z1=X*Y
5	2		LOADX,C2
13	-		LOADY,S
107	2		ZZ=X+Y
2	1		LOADX,Z1
12	2		LOADY,Z2

104	1	Z1=X/Y
201	1	R=Z1
Code	1 - 10	load X
Code	11 - 20	load Y
Code	101 - 110	operation
Code	201	determine least square

The performance of the interpreter would be even better when the operations are performed by an assembly routine using a mathematic coprocessor.
Line 12550 prints the current fit cycle (LOOPCOUNT% + 1), and the time the result is expected (STUNDEN%, MINUTEN%, SEKUNDEN%). On the next line, the lowest value of the search area for each parameter is shown (LBEG()). This is followed by the largest value of the search area for each parameter (LENDC). Then, the fit control variables (FIT%( )) showing the tendency of the parameters (up = positive, down = negative, stable = 0) are displayed (FAK%()). Line 12600 prints the ultimate boundaries for the fit parameters. Default for the boudaries is 0 + epsilon for the lower and machine infinity for the upper boundary.
FIT_UPDATE2 procedure (lines 12700ff)
This procedure shows the intermediate results nvar(5,) - nvar(14) for all parameters and the current minimal root mean square (ROOTSQ).
INTP_SIMU procedure (lines 18000ff)
This procedure loads the equation into the active interpreter string (INTP$). It loads the pointer for the array length (AN%) from the header of the register and clears the intermediate buffers (zr(i)=0). Then it loads constants 1 to 6 with the parameter values (subroutine on line 18700), and it loads the X-values (S) into one of the intermediate buffers (dy(2,i)).

Then the simulation is executed by calling the INTERPRETER procedure. Upon return of the result of the simulation is copied into the I/O buffer re(i), so that the result can be stored in the workfile by the main SIMULATION routine.

CONV_ADDR procedure (lines 18800ff)

The fit and simulation equation has to be able to access experimental data of the source register in some cases. Because the equation needs to be applied to different source register, the absolute address can only be determined at runtime. The variable addresses are coded by typint "RR" in the equation. The CONV_ADDR procedure replaces the "RR" with the number of the source register at runtime.

INTP_FIT procedure (lines 19000ff)

This small procedure prepares the buffers and variables so that the standard mathematical interpreter procedure can be used to perform direct curve fits. That means the interpreter is used as an equation interpreter in this environment. Only minor modifications are necessary to use this interpreter as a compiler (see above).

First it loads the active interpreter string (INTP$) with the equation and sets the array length pointer (AN%) with the number of data points. Next the intermediate buffers are cleared and the constants C(1) - C(5) are loaded with the values of parameters 1 to 5. Then the interpreter calculates the theoretical curve. The result is returned in the rey() array. The next routine calculates the sum of squares of differences between the actual curve and the theoretical curve (DEVM = DEVM+(rey(i)-re(i))^2*dy(1,i)). Then the root is calculated and compared to the minimal value n (DEV). If a new minimum has been found, the parameters are copied into the header of the register (line 19766).

Chapter 5
GRAPHIC PRESENTATION

5.1 Startup and Termination

When the 'PL.EXE' overlay is started, a number of variables local to this overlay are defined. Similar to other applications the file buffers have to be declared: A buffer for the line printer is created with the open "LPT1:" command. A prerequisite is the declaration of common variables. As in all other overlays this is done in the 'COMVAR.BAS' file which is included at the beginning of the source code. The different drive strings (e.g. a: b:) are created from the DRT$ string with the IDRIVE() common variables. For that reason the system configuration can be changed by the user. The definition of variables starts with two general purpose buffers REX(), REY(). The DISLIN array is used later to convert the internal linetypes of the MICHFIT software into linetypes for the (GIOS) graphic input/output system. Next several local variables are initiated:

TSSET	= 1	first softkey Display Menu
CURSOR	= 1	type of cursor is set to data cursor
CW	= 1	it prints to the first data print
IDMAX	= 0	maximal number of draw points is zero
IDC	= 1	pointer for draw cursor is set to first position
PGA(9)	= 1	

PLOTMEN%	= 0	Plotsize Menu is not on screen
IANNOT	= 0	
BACK%	= 0	
UPPER%	= 1	the Plotsize Menu shows the upper half
TOUCHS	= 0	the touch sensitivity is switched off
TTAUS	= TAUS	temporary output device is identical to the output device
RUNNING%	= 1	the main loop is run infinitely
ISYM	= 1	the first symbol is selected
ILEER	= 5	empty space during inputs
INSYM	= 1	active symbol (selected in symgol menu)
TSHARE	= 2	the background timeshare n 90%(1/2)
IPLOT	= 1	the background counter is set
AUFSPULEN%	= 0	the software is in the direct output mode

Next the exceptional characters are defined: Lines 326 to 327 define an array which is used to remap several ASCII characters of the extended set (>128) to an exceptional character list defined in the system file. On line 330 the prompt messages for the Plotsize Menu are read from the system file. This is only necessary if they have been overwritten by an other overlay or if they have been cleared by the emergency clean string space routine. In both cases the pointer (26) is different from 1. On line 345 the first character set is loaded from the system file (symbol character set for the display). Then a set of plot functions is defined (subroutine on line 20000). If the graphic overlay is called in the autoscale mode and in the 'zoom off' mode (AUTOSCALE = 1, POINTER% (21) = 1, POINTER% (23) = 0 and the display is not locked, the display is cleared before the autoscaling procedure is called. If the graphic overlay is called with the INTERPOLATE flag set, additional data points are generated between existing data points if deltaY between the data points exceeds 1% of the scaling. This procedure gives a smoother appearance to plots drawn with lines. Before the interpolate procedure is called the procedure SMOOTH with the SMOOTH intensity 2 is executed.

The graphic overlay can be called to show the Plotsize Menu PINIT = 1 or to perform graphic output PINIT = 2. In the second case the routine for the selected output device (TAUS) is called and the graphic subroutine is left again (RUNNIN% = 0). In the first case the Plotsize Menu is shown and the main program watches for user inputs, and outputs data to the plotter. For lack of memory the command - and mathematic - interpreter have not been included into this program.

If the 'zoom' mode is off and the display is not locked, the active plot has to be cleared. (line 1000). To do this the display data procedure is called with the pointer set to erase (POINTER%(22) = 0). After that the graphic screen is erased and the alphanumeric screen is shown. Because of problems of the Microsoft BASIC in handling string arrays, the string array elements 1-5 have to be copied to the strings XBES$, YBES$, XOBES$, YRBES$, and PTITEL$. The reverse copy operation is performed by a subroutine on line 19000. Next, the menu mask of the next menu and the appropriate softkeys are shown on the display. Lines 1050, 1070, 1091, and 1092 remove unnecessary strings from the stringspace and pack the rest together (garbage collection). After that the active menu (IME) is stored in the common area (POINTER%(19)). The same applies to the maximum number of draw points (IDMAX). On line 1100 the program that has called the graphic presentation overlay is reactivated (HAIN PRET$).

IN2NUM	procedure (lines 3960ff) see VKI
BACK_OUT	procedure (lines 4300ff) see VKI
CLOSE_SPOOLFILE	procedure (lines 4400ff) see VKI
CHECK_INPUT	procedure (lines 5000ff) see VKI
ERROR_RECOVER	procedure (lines 6000ff) see VKI
SET_TOUCH	procedure (lines 8500ff) see VKI
KEYLABEL	procedure (lines 8700ff) see VKI

FIND_OUTPUT procedure (lines 7000ff)
This procedure calls the appropriate procedure for each value of the TAUS variables:

1	Display	Calls Display Menu on line 2200
2	Plotter	Calls plot procedure on line 25000
3	Printer	Calls print routine on line 24000
4	Spoolfile	Calls plot routine put redirects output to a spoolfile To do this the number of files in spool pointer (POINTER%(1)) is incremented and a file name is generated. The file is opened and the redirection

	flag (AUFSPULEN%) is set.
Plotfile	For this function the data handling subroutine has to be reactivated.

5.2 Definition of Functions

This subroutine defines several BASIC functions that ease the access to graphic facilities of the BIOS. The graphic cursor is placed on the fourth data point (cw=4) of the first data curve (cursg=1). The variables X% and Y% define the size of the zoomed image in display pixels.

The DISLIN%() array is initiated so that the internal linetpyes of the MICHFIT software are converted into filetypes of the graphic input/output system. Several strings are loaded with escape sequences so that they can be printed to the terminal later on:

Escape string	function	escape sequence
TOGGLEM$	set toggle mode	(ESC)m3A
GT$	set graphic text	(ESC)*dS
AT$	set alphanumeric text	(ESC)*dT
GROFF$	switch off graphic display	(ESC)*dD+(ESC)*dT
CLEARM$	clear mode	(ESC)*m1A
SETM$	set mode	(ESC)*m2A

A set of functions is used to convert user units into pixels and to perform display or plot functions:

FNPLU$(X,Y)	move to position X,Y in user units with pen lifted on display
FNPLD$(X,Y)	move to position X,Y in user units with pen lowered

FNPMU$(X,Y)	move to position X,Y in user units with plotter pen lifted
FNPMD$(X,Y)	move to position X,Y in user units with plotter pen lowered
FNPLU2$(X,Y)	!
FNPLD2$(X,Y)	! same above but in protrait mode
FNPMU2$(X,Y)	! instead of landscape mode
FNPMD2$(X,Y)	!
FNXW(X)	converts user untis into pixel units
FNYW(Y)	converts user units into pixel units
FNSCX(Xmin,Xmax)	define user to pixel
FNSCY(Ymin,Ymax)	transformation
FNCURS$(X,Y)	move graphic cursor to position X,Y
FNW$(X,Y)	converts user untis into plotter units (ASCII)
FNV$(X,Y)	converts user units into plotter units (ASCII)

5.3 Plotsize Menu

PLOTSIZE_MENU procedure (lines 21000ff)

This procedure reads the content of the screen mask file 4, cleans the display and copies the mask from the I/O buffer to the alpha memory (line 21020). The old touchfields are cleared and touchfields are setup by the subroutine on line 30000. The menu update pointer points to the first update field. Then field that could have been by the data handling subroutine are selected for the next update. The new softkey are labelled and the PLOTMEN% pointer is set to indicate that the Plotsize Menu is in the alphanumeric screen memory.

```
Title of Plot:       sample title
Output destination:  display         Pen Title/Axes:       2
```

```
                        1
Position of Plot:       along          Annotate
Lines/Pen:              1              1

              X-  Y-  left lower speed tick% Line S. Col.W
            and size       rim
 Plot size 20   15    3    2     20     1      10      0

            min.  max.  step  in. step       axes text
X-lower 7    0     23    2.5       0       cytochrome c

Y-left 7    .7   1.09   .05        0      turnovernumber

X-upper 3    0    23    2.5        0

Y-right 3   .7   1.09   .05        0

            title  axes   digits  symbol    annotate
X-size       .3     .25    .07     .15        .2
Y-size       .4     .13    .11     .02        .25


               title  X-lower  Y-left  X-upper  Y-right
X-coord. text   50      50      -10      50      108

Y-coord. text  108     -10       50     105       50

X-Pos. digits   1       -2      -5.5     -2       .5

Y-Pos. digits   1       -1      -.25     .25     -.25

#  dir     X-pos.   Y-pos.      text
1  1      +10.00   +50.00     text of annotate line 1
```

2	1	+10.00	+40.00	text of annotate line 2
3	1	+10.00	+30.00	text of annotate line 3
4	1	+10.00	*20.00	text of annotate line 4

PLOTSIZE_UPDATE procedure (lines 21025ff)
Each time the PLOTSIZE_MENU update procedure is called, it checks one small part of the screen and updates if necessary. Only the half (upper/lower) that appears on the screen is actually updated (lines 21032 and 21033). The on X GOSUB commands on lines 21040 and 21050 are used to select the update procedures.
Upper part

1	21200	PTITEL$	plotfile
2	21210	TAUS	output device
3	21220	pg(11),pg(12)	pen for title and axes
4	21230	pg(13)	landscape/portrait and layout of plot
5	21240	pg(14),pg(15)	number of annotate lines pen used for annotate
6	21250	pg(1),pg(2)	X and Y size plot in cm
		pg(3),pg(4)	left and lower rim of plot in cm
		pg(5),pg(6)	speed of pen, tick size
		pg(7),pg(8)	line spacing, column width
7	21260	pxu#(),xbes$	definition of lower X-axis
8	21270	pyl#(),ybes$	definition of left Y-axis
9	21280	pxo#(),xobes$	definition of upper X-axis
10	21290	pyr#(),yrbes$	definition of right Y-

			axis
11	21300	pgx()	X sizes of labels in cm
12	21210	pgy()	Y sizes of labels in cm

Lower part

13	21320	pgx(i)	X-coord.text
	21230	pgy(i)	Y-coord.text
	21240	pgx(i)	X-pos digits
	21250	pgy(i)	Y-pos digits
	21260	BESCHR$	annotate lines (4)
	21270	title line	for upper block
	21280	title line	for lower block

1 = title
2 = x-axis
3 = y-left
4 = x-upper
5 = y-right

PLOTSIZE_CMD procedure (lines 21400ff)
Similar to the command procedures for the other menus the plot commands are divided in three groups: Single field commands, field array commands, and softkey commands. Because the field sizes have not yet been stored in a systemfile, the type of field has to be decoded from the returned character (A$). On lines 21400 to 21420 the touched array (1-5) is evaluated. The variable A on line 21425 identifies a touched softkey, it is identical in function to the TSOFT variable in the other menus (A = ASC(a$) - 17). The softkeys provide return codes in the range from 18 to 25 when they are touched. The variable ii on line 21440 is identical to the TFELD variable. It is determined from the position of the returned character (A$) in the code string H1$. On line 21430 the store menu flag (ip(90)), the clear command line flag (ip(89)), and the

Plotsize Menu update pointer are set. Then, if an array field has been touched, one of the array field routines is activated.

1	21460	modify title of plot	- remove title from graphic screen in zoom/off mode - modify PTITEL$ - set update pointer
2	21740	change pen of title and axes	- remove title and axes from graphic screen in zoom/off mode - pg(11) = pen of title pg(12) = pen of axes - set update pointer 63
3	21480	change number of annotate lines and pen	- remove annotate lines from graphic screen in zoom /off mode pg(14) = no of lines pg(15) = pen - set update pointer 65
4	21490	change output device	- TAUS - set update pointer 62
5	21500	modify X-axis (lower)	- clear axis from graphic screen in zoom/off mode
6	21510	modify Y-axis (left)	- load axis control variable

			pxu#(5) - pyr#(5)
7	21520	modify X-axis (upper)-	
8	21530	modify Y-axis (right)-	set update pointer 67-70
9	21540	change from landscape to portrait	- increment pg(13) - clear screen in zoom off/ mode - set update pointers 64, 66, and 71-76. - set update flags 91-99 for graphic screen

The axis control variables pxu#(5), pyl#(5), pxo#(5), and pyr#(5) are used to change the appearance of the axes according to the following scheme:

1	draw ticks
2	draw digits
4	draw labels
8	switch to logarithmic display

The axis control variable for an axis with ticks, digits and labels has the value 7 (1+2+4).
On line 21600 the TCOL and TROW variables for the first three field arrays are evaluated. The second field array modifies the entries for the four axes: The first column modifies the minima, the second modifies the maxima, the third column modifies the increments, the fourth column modifies the first increment, and the fifth column modifies the labels (lines 21642-21645). The first field array modifies the size of the plot, the position of the plot the plotter speed and a few other variables (pg(1) - pg(7) and pga(8)). Again if the display is shown in the zoom off mode, the active

plot has to be removed from the graphic screen (line 21620). Lines 6 and 7 of this array are used to modify the X-and Y sizes of the labels (title, axes, digits, symbols, annotate).
The routines for the fourth field array can be found on lines 21650 to 21678. Again, the first two lines are used to determine the TROW and the TCOL variables. Then, in the zoom off mode the title axes and annotate lines and the corresponding update flags are cleared. Lines 21669 and 21670 input the positions of the labels in % of the plot frame. The labels are centered around the given position. Lines 21675 and 21677 input the position of the digits relative to the origin in units of character size.
The routine for the fifth field array can be found on lines 21668-21696. After the TROW and TCOL variables have been calculated, the annotate lines are removed from the graphic screen if it is in the zoom off mode. Column 1 is used to modify the direction of the labels (1 = horizontal 2 = vertical a.s.o.), column 2 and 3 are used to move the label around (the position is given in % of the frame size). Column 4 is used to modify the annotate text. Before the routine stops, the update pointers (81-84) are set.

1 21720 positive/negative

This softkey is used to select between the positive display mode (dark lines on light screen, POINTER%(24) = 1) and the negative display mode (light lines on dark screen POINTER%(24) = 0). To do this, the SETM$ and CLEARM$ escape sequences are changed by the subroutine

		on line 20500. Then the screen is cleared and the update flags are reset.
2	21740	autoscale This softkey calls the autoscale procedure (line 27000). It sets the update pointers 67 to 70.
3	21760	next position This softkey increments the position pointer (pg(9)). The new position values are provided by the DEF_POS procedure on line 29500.
4	21780	execute output The output is sent to the selected output device (line 7000).
5	21800	center text This routine enters annotate lines as a block of text. It inputs lines (max 15) until an empty line is found. The position of the line is calculated from the coordinate, the line number and the line spacing (pg(7)).
6	21820	plot spool Commands can be given to the plot spool (see SPOOL procedure in VKI overlay).
7	21840	upper lower The display can flip from the upper half to the lower half or via versa. This is decided by the value of the UPPER% flag.
8	21860	roll annotate

5.4 Display Menu

The Display menu breaks up the task of showing a plot into several subtasks. These tasks are executed by the

DISPLAY_UPDATE procedure in a similar manner like the alphanumeric menus. When the Display Menu is acitivated, the menu pointer (IME) is set and the first set of softkeys is activated TSSET = 1). The alphanumeric screen is switched off and graphic text mode is selected. The graphic display is operated in the TOGGLE mode, that means writing a particular entry once sets the display pixels and writing the same entry a second time clears it again. This is used to perform selective erases so that several plots can be maintained on the graphic screen without generating them new all the time ('zoom off' mode). The software has to maintain a memory about which entries are on the screen so that they can be written a second time before they are changed (IP 90-99). Line 22020 divides the task of showing one plot into 10 subtasks (BLOCK% = 1 to 10). After that the softkey labels are shown by the procedure on line 22100. If the DISPLAY LOCK flag (POINTER%(23)) is set the position of the plot has to be changed before a new plot can be shown.

DISPLAY_CMD procedure (lines 22400ff)
This procedure distinguishes among 6 sets of softkeys (line 22440). They are arranged in a tree-like fashion: Softkey set 1 is the root set which can activate 5 subsets of softkeys.

1	Root softkeys	22500
2	More softkeys	22900
3	Annotate softkeys	22800
4	Draw softkeys	22700
5	Hardcopy softkeys	22600
6	Select symbol	22650

5.4.1 Root Softkeys

Softkey 1	'Unlock Display' (line 22510) clears the internal flags for the display (ip(91) - ip(99)) so that the new plot can be drawn. This is used if two plots have to be shown on the same position (more than 10 data curves) or if an inset is added to an existing plot.
Softkey 2	'Draw' (line 22520) activates the draw softkeys (TSSET = 4). The new labels are shown by the LABEL_SK procedure on line 21100.
Softkey 3	'Select Symbol' (line 22530) activates the Select Symbol Menu (TSSET = 6). Symbols 1 to 40 (see thesis) are shown in the area where softkey labels appear. The symbols are selected by means of the graphic cursor when the key 's' (select) is pressed.
Softkey 4	Plotsize Menu' (line 22540) changes to the Plotsize Menu.
Softkey 5	'Output Menu' (line 22550) changes to the Output Menu.
Softkey 6	'Annotate on' (line 22560) activates the annotate softkeys (TSSET = 3). The new labels are shown by the LABEL_SK procedure on line 21100. The command line is cleared using the subroutine CLEAR_CMDLIN.
Softkey 7	'Hardcopy' (line 22570) activates the hardcopy softkeys (TSSET = 5). The message *'Please select output device'* is shown on the command line just above the softkey labels.

Softkey 8	'More Softkeys' (line 22580) selects the more soffkey set (TSSET = 2).

5.4.2 Hardcopy Softkeys

The hardcopy softkeys (lines 22600-22645) are used to sent the plot currently on the screen to one of the output devices (Plotter, Printer(table), Printer (screen dump),file, spoolfile). Softkey 1 and 8 both exit back to the root softkey set.

Softkey 2	'Plot' activates the PLOT procedure (line 25000). The plot is sent to the plotter immediately. The plot can be aborted by pressing the 'a' key. A plot which has been plotted is locked to the position of the layout (POINTER%(23) = 1).
Softkey 3	'Print table' activates the PRINT_TAB procedure (line 24000). This procedure prints a table of the data curves.
Softkey 4	'Graphic Dump' dumps the screen to the printer. This is performed by the escape sequences "(ESC)&8p755dF" and "(ESC)&p3D".
Softkey 5	'File' stores the information that was needed to generate the current plot in a file. This is done with the WRITE_PLOTFILE procedure of the data handling subroutine.
Softkey 6	'Spoole' redirects the output of the plot procedure to a file. Several files can form a queue which is then output to the plotter in the same sequences as they were added by the SPOOL procedure. Before the PLOT procedure is started, a

brake point can be inserted at the beginning of the file. In this case 'STOP' is written to the file (line 22637). The file name of the spool files is generated from PLOT and the queue position number (1=A 2=B a.s.o.). Then the screen size variables are saved and the PLOT procedure is started with the AUFSPULEN% flag set.

5.4.3 Select Symbol

The Select Symbol Menu shows all available linetypes, symbols and area fills in the lower part of the screen. The cursor is allowed to move free on the entire screen (CURSOR = 11). It can be moved by the graphic cursor move keys in the graphic keypad. If 's' is detected, the cursor position is checked if it was on one of the symbols shown by the SYMBOL_MENU procedure. The cursor has to be within an area of +/- 40 pixels horizontally and +/- 5 pixels vertically so that it is detected by the routine on lines 22660-22672.

The SELECT_SYMBOL Menu is left when one of the softkeys is pressed (line 22656) in this case the lowest part of the display is cleared (subroutine on line 22695 uses the escape sequences: '(ESC) *m 0 0 511 65' and '(ESC) *m1G'. The same approach is used by the other partial clear routines (lines 22380 and 22390).

The SYMBOL_MENU procedure (lines 22673ff)

This procedure shows the message *'select with cursor (5 = select exit: any softkey)'* on the command line. Below this line 5 rows with each 10 symbols or linetypes are displayed. Line 22684 shows the user definable symbols, line 22685 shows small pieces of lines with the different linetypes. Line 22686 shows the symbols out of a fixed character set. The rest of the fields shows the symbol curve (e.g. S25, line 22687).

5.4.4 Draw

The DRAW softkeys (lines 22700 - 22795) offer a limited set of functions to draw lines and symbols anywhere on the display. This is done by moving the graphic cursor (CURSOR = 11, line 22702) to the desired location and selecting one of the softkeys.

Softkey 1	'Enter Penup' (line 22710) lifts the pen and moves from the last location to the current position of the graphic cursor (IDPC = 0 lift pen). Each time a new entry is added, the lines are selectively erased (gosub 23700) and then redrawn (gosub 23700).
Softkey 2	'Enter Pendown' (line 22720) performs the same function but with the pen lowered, that means a line is drawn from the last location to the current position of the cursor.
Softkey 3	'Move cursor' (line 22730) moves the graphic cursor from one stored point to the other. If the last point has been reached it jumps back to the first point.
Softkey 4	'Delete' (line 22740) uses the pointer of softkey 3 (IDC) to erase one draw point. Before this is done the draw points are removed from the display (gosub 23700). Then the points above IDC are copied one location downwards (line 22742) and the last point is cleared. The cursor is positioned on the new

	point with the array position IDC and the draw points are redrawn (line 22744).
Softkey 5	'Redraw' (line 22750) redraws the current plot.
Softkey 6	'Enter Symbol' (line 22760) draws a symbol with the active symbol type ISYM at the location of the graphic cursor.
Softkey 8	'Exit' exits from the DRAW softkeys.

The routine on line 22785 adds one entry into the draw array. Before this can be done, the position of the graphic cursor which is returned in pixel units has to be converted into relative units compared to the frame of the plot (0-1). These units are multiplied by 10'000 so that they can be stored in integer variables (line 22790).

5.4.5 Annotate

The ANNOTATE softkeys (lines 22800-22895) are used with the graphic cursor (CURSOR = 11) to place a label anywhere on the plot.
Softkeys 1 to 3 are used to change the title. The label of the X-axis and the label of the Y-axis.
Softkey 6 is used to change back to the Root Menu.
After Softkey 7 has been pressed, the labels are removed from the screen and the prompt *'Text?'* is shown. Several lines of text can be entered in a block (1 up to 15) until no character is grown on one line. A limited set of commands can be used to chage the line spacing (code &A line 22873), or the direction (code &R line 22874). On line 22876 the number of annotate lines is incremented, direction, positions, length and the label itself are stored. On line 22879 the update pointers for the Plotsize Menu are set.

Softkey 8 removes all annotate labels by setting the number of lines pointer pg(14) to zero.

5.4.6 More Softkeys

The MORE softkeys (lines 22900-22990) provide additional functions to edit the graphic display.

Softkey 1 'Smooth Display' (line 22910) erases selectively one data curve and redraws it after smoothing (subroutine on line 27500).

Softkey 2 'Interpolate Line' (line 22920) erases selectively one data curve and redraws it after it has been processed by the INTERPOLATE procedure (line 26500).

Softkey 3 'Replace Symbol' (line 22930) again selectively erases one data curve and redraws it with the new line/or symbol type (as selected in the Select Symbol Menu) (PSYM() = INSYM).

Softkey 4 'Replace Pen' (line 22940) performs a similar function as softkey 3 apart from the fact that the pen type has to be entered after the prompt. All functions proceed similarly: First, the command line is cleared (subroutine 22390), then the graphic cursor is positioned on the command line (subroutine 22350) and a BASIC INPUT is performed. Then the curve is removed (subroutine 23300), the function is performed on the data, the curve is drawn againg, and the command line is cleared.

Softkey 5 'Lock Graph' (line 22950) locks the display manually (POINTER%(23) = 1).

Softkey 6 'Clear Screen' (line 22960) clears the screen amd redraws the plot which is currently active.

Softkey 7 'Zoom on / off' (line 22970) selects between 'zoom on' mode POINTER%(21) = 0 and 'zoom off' mode POINTER%(21) = 1. In both cases the display has to be cleared and the actual plot is redrawn with the new scaling.

Softkey 8 exits to the first softkey set.

5.5 Display procedures

Building up a plot on the display is split up into several procedures which are called one after the other by the DISPLAY_UPDATE procedure:

```
      DISPLAY_SCALE
(4)   DISPLAY_AXES
      DISPLAY_ANNOTATE
      DISPLAY_FRAME
(10)  DISPLAY_DATA
      DISPLAY_COLUMN
      DISPLAY_DRAW
```

DISP_SCALE procedure (lines 23000ff)

During the time a display is generated the keyboard is monitored periodically and at the same time a background output to the plotter is done every fourth cycle (TSHARE = 4). Possible errors would cause the termination of display generation and the program would resume with the main program (ONERROR GOTO 21900, RESUME 500). The subroutine CHECK_AXES is called to detect invalid axes (STEP>=0). On lines 23005 and 23025 the size of the frame is determined for the 'zoom on' mode (POINTER%(21) = 0). The frame is determined by the lower left (PXMIN, PXMIN) and the upper right corner (PXMAX, PXMAX) in pixels. The proportion of the plot (Y/X, PG(2)/PG(1)) is maintained by modifying the upper right corner either in X-(H4>1) or in Y-direction (H4<1). In the 'zoom off' mode the frame is calculated using the X-size (PG(1)) and left rim (PG(3)) and the Y-size (PG(2)) and lower rim (PG(4)). In the portrait mode (PG(13) = 2 or 4) the corners have to be calculated as follows:

```
        PXMIN = left rim * 15 + 100
        PXMAX = (left rim + X-size) * 15 + 100
        PYMIN = 390 - (X-size + lower rim) * 13
        PYMAX = 390 - (lower rim) * 13.
```

The offset of 100 in the PXMIN and PXMAX is due to the fact that an A4 sheet of paper cannot use the entire screen in portrait mode. If the pen for the axes is zero (PG(12)), the screen is not cleared, this is normally used to draw more than 10 data curves to the same plot. In the 'zoom on' mode the display is normally cleared (line 23035). If display is in negative mode (POINTER%(24) = 1), the entire screen is set. In portrait mode (zoom off). Only the area occupied by the A4 sheet is set.

DISPLAY_AXES procedure (lines 23050ff)

The size of the graphic text is adjusted with the SIZE procedure. The scaling functions FNSCX and FNSCY are called to convert the user units into pixel units. F5 and F6 are set to 1% of the frame size in user units. Then the XAXIS procedure (subroutine on line 25900) is called. This routine draws a horizontal line with ticks and digits with increments DEL# and the first increment DELO# from the MIN# to the MAX# in user units. The coordinates of the starting point are defined by AYPOS and AXPOS. The direction of the labels is set by the DIR procedure. The relative positions of the labels are defined by ZXPO and ZYPO, the tick size is defined by F6 and AOP defines the type and appearance of the axis. The Y-axis is drawn by the YAXIS procedure (subroutine on line 25950) in a similar way (lines 23066-23069).

The scaling of the axes is repeated for the X-upper and Y-right axes and these axes can be drawn with different units (lines 23070-23081). On line 23082 the scaling is changed to X-lower and Y-left units again. The X-upper axis can be drawn using the user units from the X-lower axis simply by selecting a negative number for the PXO#(5) variable (line 23065). The same applies for the Y-right axis (line 23068) and vice versa (line 23077 and 23081).

DISPLAY_ANNOTATE procedure (lines 23090ff)

The scaling is changed to relative units (0-100%), the size of the labels is set by the SIZE procedure. PG(14) lines are written to the display with the directions PG(i+15). The FNXW and FNYW functions are used to convert the relative units into pixel units. The text is printed after the PLOTU function has been called.

DISPLAY_FRAME procedure (lines 23100ff)

Again, a relative scaling (0-100%) is used to display the frame, the title, and the labels of the axes. The character size for the labels are taken from the PGX(2) and PGY(2) common variables. The variables F3 and F4 contain the relative character sizes in % of total frame size, SL contains half the length of the label. The start position of the label is determined by the relative position of the labels (PGX(7) and PGY(7) for the X-lower axis), and by half of the label length times relative character size (line 23102). If the third bit of the axe control variable PXU#(5) is set, the label is printed. The same procedure applies to the other axes (lines 23109-23112) and for the title (line 23122). The frame is drawn using a solid line (LINETYPE%(1)) with relative position (0,0 - 100,100) (lines 23115-23119).

DISPLAY_DATA procedure (Lines 23250ff)

Data are shown using the user units from the X-lower and Y-left axes. The FOR-NEXT loop on line 23270 calls the DCURVE procedure and monitors the keyboard if a pen different from zero is selected (STIFT%()<>0).

DCURVE procedure (lines23300ff)

The size for the symbols is adjusted using the PGX(4) and PGY(4) common variables. The procedure is left if either the number of data points is lower than 3 (DMAXC) < 5) or the symbol type is negative or zero (PSYM() <1). If either the number of data points or the symbol type are too large, they are set to the largest valid value. If a symbol in the range from 1 to 10 is selected, the user definable character set has to be read in (subroutine on line 26400).

If a line is selected ISYM > 10 and ISYM < 21 the linetype is defined (subroutine on line 26100). If the first value exceeds the plot limits, it is set to PXMIN, PYMIN (line 23332) with the PLOTU command. The loop on lines 23350 to 23380 plots the data points to the screen. No data point is shown if it exceeds the plot limits (lines 23352 and 23354). The data is then converted to pixel units and again checked for overflow (lines 23368-23372). Then depending on the ISYM variable either a line, (line 233373) or a user definable symbol (line 23375), a column (line 23377 subroutine on line 23800), or an ASCII character (line

23378) is drawn. After the last data point has been drawn, the pen is lifted and the procedure is left.
CHECK_AXES procedure (lines 23600ff)
This procedure prevents that the display axes procedure 'hangs' when the STEP variable is selected too low so that more than 99 ticks and digits are drawn.

DISPLAY_DRAW procedure (lines 23700ff)
This procedure plots lines and symbols to the display as they have been input by means of the DRAW softkeys. It uses the subroutine on line 22795 to convert the stored position into pixel units. If a symbol or a column has been selected this is displayed using the subroutine on line 23770. A linetype is selected when the IDP() variable is in the range from 10 to 20. In the pen down mode the IDP() variable holds the linetype and the line is drawn with the PLOTD command. In the pen up mode the IDP() variable is zero, here a move is performed with the PLOTU command.
DISPLAY_COLUMN procedure (lines 23800ff)
The column width is determined using the column width variable PG(8) and the X2 variable (0,5% of frame size). The X1 and Y1 variables are passed to this procedure, they define the center and the height of the column.

After the frame has been drawn (line 23810 and 23815), the area is filled. The area is hatched vertically (subroutine on line 23900, ISYM = 22) or hatched horizontally (subroutine on line 23920, ISYM = 23). With symbol type and vertical hatching routines. The area is hatched with sloped lines using escape sequences (ISYM 25-27, see GIOS manual).The escape sequence on line 23855 perform a complete area fill

(ISYM = 28). With ISYM set to 29 the column is vertically hatched with dotted lines, and with ISYM set to 30 the column is vertically hatched with a denser spacing.

5.6 Print procedures

PRINT_HEADER procedure (lines 24000ff)

This procedure prints the title , the labels of the axes, and the annotate lines on the line printer.

PRINT_DATA procedure (lines 24080ff)

The data is printed in two columns using the format string FORMP1$. If a number is larger than 9999 or smaller than 0.01 it is printed using the floating point format "#####^^^^^".

5.7 Plot procedures

The generation of a plot on the plotter is split into several subtasks:

PLOT_FRAME	PLOT_ANNOTATE	PLOT_DRAW
PLOT_AXES	PLOT_DATA	
	PLOT_COLUMN	

PLOT_FRAME procedure (lines 25000ff)

First the axes are checked by the CHECK_AXES procedure. Then the I/O buffer for the plotter is opened. The ERROR procedure on line 6000 is called in case of an error. The frame is arranged using the X-size, Y-size, left rim, and lower rim variables. Centimeters are converted into PLOTTER units by a multiplication with 400 (line 25020). The plotter is initialized with plotter units: PL$ = "IN; IP 0,0,10900,7650," and a relative scaling (0-100) is performed with the FNSCX and FNSCY functions. The frame is plotted twice with an offset of 0.1% (lines 25050 and 25060). The lines are drawn by the function MOVE (subroutine on line 26000)

the parameter IPEN is set to 1 for pen down and 0 for pen up.
On line 25100 the character size for the title (PGX(1). PGY(1)) is sent to the plotter. The pen is in MOVEed to the title position, the direction of the label is set (depending on landscape or portrait mode pg(13)), and the plotter is switched to center label mode. Then the title is plotted using the PLOT_LABEL procedure on line 25820. The same set of commands is used to plot the labels of the axes (lines 25117 to 25144).
PLOT_ANNOTATE procedure (lines 25150ff)
The PLOT_ANNOTATE procedure sets the character size of the annotate labels by sending an "SI <X-size>, <Y-size>" command to the plotter. The variables PGX(5) and PGY(5) are converted into ASCII strings with the function STR$(). In the main loop of this procedure the labels are positioned with the MORE procedure and the direction of the label is set according to the direction variable PG(I+15) and the variable PG(13) (landscape or portrait mode). The string is then plotted with the PLOT_LABEL procedure.
PLOT_AXES procedure (lines 25192ff)
The "SP"+STR$(PG(12))+"," string is used to select the appropriate pen for the axes. If PG(12) contains zero, no axes are plotted. The scaling is set to user units of the X-lower and Y-left axis and F5 and F6 are set to 1% of frame size in user units. The character size is set using the variables PGX(3) and PGY(3). The axes are plotted by the same XAXIS and YAXIS routine like for the display. (see DISPLAY_AXES procedure and XAXIS procedure for the meaning of the parameters).

PLOT_DATA procedure (lines 25600ff)
Before the data is plotted, a mask is laid on the plot that prevents lines or symbols being plotted outside of the frame ("IW" command on line 25601). The character sizes for the symbols and the pen is set on line 25605. The main loop plots up to 10 data curves with the symbol INSYM = PSYM() and the pen STIFT% (lines 25610-25770). On line 25620 the pen is changed if the new pen is different from the pen used for the last curve. If the plot character set for the user definable characters has not been loaded, (POINTER%(25) <> 3) this is performed by the READ_CHARSET procedure on line 26400. The linetype is selected and the plotter pen is lifted and moved to the first data point (line 25650). Then the data points are plotted with the loop from line 25670 to 25750. Line 25680 lifts the pen to move to data points if they are out of range. Line 25690 skips data points if they are far outside of range. The routines on lines 25730 to 25747 plot symbols (INSYM < 11), or (lines INSYM > 10 + INSYM < 21), or columns (INSYM > 20 and INSYM < 31), or ASCII characters (INSYM > 30 and INSYM < 41). In the line plot mode the pen is lifted after the last point has been plotted. After having finished all data points of all curves the pen is stored and the plotter moves to the origin (line 25780). The I/O buffer is closed and the procedure is left.
SENDPL procedure (lines 25800ff)
This small procedure adds string delimiters (chr$(34)) to the PL$ string before it is printed to the plotter.
PLOT_LABEL procedure (lines 25820ff)
This procedure takes character per character from the annotate label (H5$) and compares then with the superscript code "@", the subscript code "#" and the

exeptional character code "&". Superscript and subscripts are plotted with 75% character size (subroutine on line 25830 and 25832). ("CP 0,"+STR$(CYOFF)) ("SI"+STR$(AKTSIX) + ","
+STR$(AKTSIY)). This subroutine increments the character counter plots one character and switches back to normal mode. IF the exeptional character is found in the exeption list the subroutine on line 25835 checks if the character set is loaded and plots the character. (See system files for exceptional characters.)
PLOT_DRAW procedure (lines 25850ff)
The stored positions IDX() and IDY() are converted into relative positions (0-100%). A symbol is drawn if an IDP() value larger than 100 is found. A new linetype is selected (CALL LINETYPE (L%)) if IDP is between 11 and 20. If IDP is 1 the pen is lowered and if IDP is zero the pen is lifted.
XAXIS procedure (lines 25900ff)
The tick on/off flag AT%, the digit on/off flag A2%, and the logarithm flag LG% are masked out of the AOP variable. If the AX% variable is zero or more than 99 labels have to be plotted, the procedure is left. The ticks and labels are plotted using the FOR-NEXT loop from lines 25910 to 25935. The K# variable contains the position of the label in user units.

In the linear mode LG% = 0 the tick is plotted with the function MOVE on line 25920. On the display the character offsets are performed on line 25930 before the string is printed STR$(K1). On the plotter the offsets are performed by the "CP" command. The label is printed with SENDPL procedure after the number has been converted into ASCII. Due to rounding errors zero is expressed as a very small number (e.g. 1.75 * 10 E-21) this is corrected on line 25929: If K# is very small compared to del# K1 is set to zero.

YAXIS procedure (lines 25950ff)

This procedure is identical to the XAXIS procedure except for the fact that ticks are drawn horizontally and the labels are spaced vertically. Source code lines are exactly 50 lines higher than for the XAXIS procedure.

MOVE procedure (lines 26000ff)

This procedure moves the display and the plotter pens. It is called with five parameters, the output device (TTAUS = 1 = display, TTAUS = 2 = plotter), the pen state (IPEN = 0 = lifted pen, IPEN = 1 = lowered pen), the transformation (PG(13) = 1 or 3 = landscape, PG (13) = 2 or 4 = portrait) and the X/Y data in H1 and H2. Depending on the pen state and the transformation four different plot functions are used (FNPMU$, FNPMU2$,FNPM$, and FNPM2$). The display part selects between the PLOTU and PLOTD calls depending on the pen state. Before displaying, the user X/Y data is transformed to pixel X/Y data with the FNXW and FNYW functions.

LINETYPE procedure (lines 26100ff)

This procedure adjust linetypes of plotter and display to the internal linetypes of the MICHFIT software. The linetype of the plotter is selected with the "LT, (linetype), (size)" command. The linetype of the display is selected with the LINETYPE assembly language function after the types have been converted using the DISLIN%( ) conversion table.

READ_CHAR_SET procedure (lines 26400ff)

This procedure opens the HSY system file and reads one of several character sets of 10 characters each. The character sets are stored in the SYM$(1-10) array internally.

INTERPOLATE procedure (lines 26500ff)

The interpolate procedure is applied to lines only (PSYM(11-20)) if they have less than 250 data points (DMAX() < 250). This routine generates additional data points (2) in between existing data points by fitting a parabola to three consecutive data points. The smoother appearance of the line is improved when the smooth procedure is called before INTERPOLATE is executed. First the data curve is copied into the buffer REX(), REY(). To either side one data point is extrapolated (line 26560 and 26570). The DELTAY variable is set to 1% of the Y-frame size in user units. In the main loop (26600-26700) the difference DEL between IREY(i+1)-REY(i)) is determined. If DEL is larger than 1% of the Y-scale and the array is not too large (320-j1) < (dmax (j)-i) three additional data points are generated. If only one condition is true only 1 additional data point is added. The slope H1 and the second derivative H2 is calculated on line 26620. Then the points are generated:

Point 1: = Y + (H2-H1) * deltaX'/3

Point 2: = Y + (H2+H1) * deltaX"/3 (see lines 26640 to 26660).

PLOT_COLUMNS procedure (lines 26800ff)

The variable X2 and Y2 are loaded with 1% of the X-resp. Y-frame size in user units. The empty column is plotted on lines 26810 and 26815 (see DISPLAY_COLUMNS). Depending on the ISYM variable different procedures are used to fill the area in the columns.

26900 HATCH VERTICAL
26920 HATCH HORIZONTAL
26940 HATCH VERTICAL DENSE

5.8 Miscellaneous Procedures

AUTOSCALE procedure (lines 27000ff)

First all data points of all curves are screened for the extreme values (lines 27020 to 27090) X00, Y00, X11 and Y11.

The differences RANGEX# = ABS(X11-X00) and RANGEY# = ABS(Y11-Y00) are then normalized to a value in the range from 4 to 10. The divisors XMUL# and YMUL# contain the number of digits the decimal point had to be shifted. The minima and maxima of the axes are divided by the same factors XMUL# and YMUL#. Negative numbers are made positive. Maxima are rounded to the next larger integer and minima are rounded to next smaller integer. Negative numbers are then converted back again.

The minima, maxima, step, and initial step variables are divided by the XMUL# respective YMUL# factors.

```
PXU#(1) = X0# * XMUL#
PXU#(2) = X1# * XMUL#
PXU#(3) = 1 * XMUL#
PXU#(4) = 0
PYL#(1) = Y0# * YMUL#
PYL#(2) = Y1# * YMUL#
PYL#(3) = 1 * YMUL#
PYL#(4) = 0
```

SMOOTH procedure (lines 27500ff)

This procedure has been described in the VKI overlay.

CURSOR procedure (lines 28000ff)
The CURSOR procedure distinguishes among three different types of graphic cursors. First a DATA CURSOR that jumps from data point to data point on one curve or among the ten data curves (CURSOR = 1-10). Second an XYCURSOR that is used to move in thr DRAW and ANNOTATE modes (CURSOR = 11) and third and XYCURSOR that prints the user units on the command line (CURSOR = 12). If the DATACURSOR hits the ends of the data curve (CW<3 or CW>dmax(cursor)) movement is inhibited (lines 28010 and 28012). In the slow mode (OMIT% = 0) the X- and Y-values of the cursor location are shown on the command line. The user units are converted into pixel units and the cursor is moved (MGCURS) functions. If the XY CURSOR has to report its position in user units, they have to be calculated by the conversion on line 28080. The result is printed on the command line.

DCURSOR procedure (lines 28100ff)
The graphic cursors are moved by keys on the numeric / graphic keyboard.

The numeric key "1" moves the DATA_CURSOR to the left, (CW = CW-0) the key "3" moves it to the right (CW= CW+1).The numeric key "5" jumps to the next data curve (CURSOR = CURSOR + 1), and the key "2" jumps to the former curve.
FAST_CURSOR procedure (lines 28140ff)
The subroutine on line 28140 monitors the keyboard. If a key autorepeats, this is detected by this routine and a counter is set ID=ID+1 if this continues, the increment variable IC is incremented so that the cursor moves faster. The cursor moves much faster when the

OMIT% variable is set so that no X- and Y-values are printed on the command line.

XY CURSOR procedure (lines 28200ff)
This procedure is similar to the DCURSOR procedure. The cursor is moved pixel by pixel in the slow mode.

1	move to left	X%=X%-1
2	move down	Y%=Y%-1
3	move to right	X%=X%+1
5	move up	Y%=Y%+1
0	report position	CURSOR = 12

If the increment variable IC is larger than 1 the cursor performs larger steps than just one pixel. This increases the speed considerably.
CHAR_SIZE procedure (lines 29100ff)
The character size on the screen has to be proportional to the character size on the plotter. This is achieved with the following equation.

$$\text{size} = \frac{\text{charsize}}{\text{framesize}} * \frac{\text{display frame size}}{\text{unit cell}} * 256$$

The character sizes on the display are defined in usual 4*7 character cell * size * 256. That means the character cell is expanded or compressed with a factor of SIZE/256. The size variable is not allowed to exceed limits of 64 and 2560. The readability of small characters is increased markedly when the Y-size is stretched by a factor of 2.

POSITION_DEFAULTS procedure (lines 29500ff)
This procedure changes the left rim and lower rim variables (PG(3) and PG (4)). If this routine is called with the position in layout variable (PG(9)) set, the appropriate left and lower rim values are set. This is

very convenient to move from one plot to the next in the default layouts. Layout 1 has only one position layout 2 has two positions (line 29600) layout 3 has 4 positions (lines 29700-29725) and layout 4 has 6 positions (lines 29750-29785).

LAYOUT_DEFAULTS procedure (lines 29800ff)

The plot can be placed anywhere on the A4 sheet by changing the left rim and the lower rim, the X-size and the Y-size, and the portrait/landscape flag. Access to different layouts is made easier with this default table for the four layouts (PG(13) = 1 to 4). If this routine is called with the PG(13) variable set, the appropriate plot sizes and character sizes are set.

SETUP_TOUCHSCREEN procedure (lines 30500ff)

Lines 30005 to 30055 define single touchfields (see Plotsize Menu). Lines 30060 to 30190 define touchfield arrays. The touchfield is then brought to the screen with the FNTOUCH function (line 30200). A touchfield is defined by four parameters, the start row ROW%, the start column COL%, the row increment ROWINC% and the column increment COLINC%. In addition to that the touchfield has to report a character to the application (e.g. 'T').

The field arrays are defined with two nested loops; the first loop defines the row, and the second loop defines the column. The response character for these fields is generated from a start code (89) which is incremented after each field (RESP$ = CHR$((j-8) * 4 + i + 89) on line 30070).

The touchfields are generated in the Data Handling overlay in a different way: They are read in a coded form into the common data area. The routine

SETUP_TOUCHSCREEN then generates the fields by decoding these variables (see VKI overlay).
The KEY_INPUT procedure has been described in the VKI overlay.

Chapter 6
Interpreters

6.1 Command interpreter

The command interpreter accepts ASCII strings as inputs. It compares the entries in this string with a command syntax interpreter by means of the system files. A line is sent to the operation interpreter for further processing if it cannot be interpreted by the command interpreter: The command interpreter has not been included into the version Feb. 88 of the MICHFIT software because the code would be to large to be executed on a PC with 256 Kbytes of RAM.
A closely similar command interpreter has been included into the SPECTRA software (see vsp.bas lines 4500ff). The input string is interpreted on lines 4515 to 4535. Command words can be as long as desired but they have to be distinguishable by the first four bites. If a command word has been found in the input string, the BEF% variable is loaded and the command is executed. The commands are listed in the following table:

No. Keyword	Function
1 EXECUTE	Executes output in Output Menu and executes input in Input Menu (same function as the respective softkey).

2 CLEAR	Clears output registers without clearing the number of curves pointer.
3 DEVICE	Selects source device in Input Menu: Syntax: Device n and selects output device in Output Menu.
4 NUMBER	Number of curves to be input in Input Menu and number of curves to be output in Output Menu.
5 AXIS	Syntax: AXIS 1,2,75 defines axis (1=x-lower 2=y-left 3=x-upper 4=y-right) and (1=minimum 2=maximum 3=step 4=initial step).
6 AVERAGE	Number of input curves to be averaged (number of curves has to be set to 1).
7 DEFAULT	Reads in default plot settings in Output Menu (depending on setting of mode).
8 MODE	Selects output transformation.
9 MENU	Selects active menu: 1=input 2=output 3=data handling 4=plotsize 5=help 6=curve fit. Syntax: MENU n
10 WAIT	Waits n seconds (enables background output and display update). Syntax: WAIT n
11 FORM	Inputs prompt or equation string up to 70 (see system files) Syntax: FORMAT 1, "text"
12 LRUN	Loads the macrofile from disc A: and executes it immediately. Syntax: LRUN: "DEMO 1" (macro files have e extension.MAK)

13 LOAD	Loads the macro file from disc A: so that it can be edited, stored, listed, and run. (see LRUN)
14 LIST	Prints a list of the loaded macro on the line printer.
15 STOP	Stops macro program execution in interpreter mode.
16 DISP	Moves cursor to location X,Y and displays the text or variable. Syntax: DISP 23.8 "text" / DISP 7,12,C1
17 INPUT	Prompts for the input of text or a variable. Syntax: INPUT "prompt", P1$ / INPUT "prompt", C1
18 RUN	Runs active macro program (starts execution at line 1).
19 STORE	Stores active macro in a file with the name and the extension ".MAK". Syntax: STORE "DEMO2"
20 MACRO	Inputs macro line. Syntax: MACRO 1,"........."
21 PEN	Selects pen of data curve (0 = select pen of axis) Syntax: PEN 1.7
22 SYMBOL	Selects symbol of data curve. Syntax: SYMBOL 1.5
23 READ	Read in file form axtive drive with active filetype. Syntax: READ "PLOTFILE"
24 ERASE	Erases workfile register in Data Handling Menu. Syntax: ERASE 12
25 NOP	Null operation

26 WINDOW	Set output window of output curve. Syntax: WINDOW 1,0,20
27 DEST	is used to set destination registers in Input Menu and source registers in Output Menu. Syntax: DESTREG 1,20
28 GOTO	Unconditional jump to line xx. Syntax: GOTO 12
29 NOP	Null operation
30 CASE	Conditional jump if condition is fulfilled. Syntax: CASE C%1=C%2.12 CASEP$1=P$2.12 CASEC%1=2.12
31 FOR	For next loop. with loop counter 1 to 10 and loop start. stop and interval. Syntax: FOR 1=1,0,1
32 NEXT	For next loop. Syntax: NEXT 1
33 VALUE	Assignment of constants and variables (convert strings). Syntax: VALUE 1="P%1" VALUE 1="21"
34 NEW	Clears macro space so that a new macro can be written.
35 SET	Is used to set symbol and plot variables Syntax: SET 3,5,34
36 DEBUG	Development function used to display intermediate results. Syntax: DEBUG 1 DEBUG 0
37 TSOFT	Performs same function as if user has pressed softkey. Syntax: SOFTKEY 1
38 TARRAY	Performs same function as if user has pressed array touchfield. Syntax: TARRAY 2.3 (row.column)

39 TFELD	Performs same function as if user has pressed single touchfield. Syntax: TFELD 2
40 EXIT	Stops macro program and exits from interpreter mode to normal user mode.

6.2 Mathematic Interpreter

Mathematic Interpreter (lines 20100ff)

The first routine counts all the opening and closing parentheses. It sets the pointers APOS and ZPOS to the highest level parantheses. Furthermore it points with GPOS to the equation sign and checks if all bytes are ASCII characters (lines 20100 - 20190). If the number of opening parantheses does not match the number of closing parantheses or if no equation sign has been found the message *'wrong input'* is displayed and interpretation is stopped (lines 20200 and 29900). If no parantheses have been found, the entire string is analyzed (line 20300) otherwise just the content of the highest level parantheses is analyzed (line 20400). The next block locates operators starting with the highest priority (1 Exponent L logarithm) over medium priority (* multiplication / division D differentiation) on to the lowest priority (+ plus - minus). When this block (lines 20500 - 20630) is left, the OPA variable contains the operation type and the OPO variable contains the position of the operator in the string. If no operator can be found and the paranthese level is zero (LMAX=0) the interpreter stops after sending the result to the desired destination (lines 25500ff). If the paranthese level is not zero the highest level parantheses are eliminated and interpretation proceeds at line 20010 (line 20710). If

none of these cases occur the message *'interpreter error'* is displayed and interpretation stops.
The routine starting with line 21000 searches for the operands to the right and to the left of the operator. The search to the left is carried out until an operator on equation sign or the start of the string is encountered (lines 21010 and 21020). Then the left operator is searched for a data type (with the routine starting at line 27000). The loop runs until one of 11 data types could be identified (see below). The datatype of the left operand is stored in the LREGA variables. To identifiy elements from two dimensional arrays the variables LQUE and IQE2 store row and column of the element. Next the data type of the right operator is identified in a similar manner (lines 21050 to 21080).
Next the right and left accumulators are loaded with data (line 22000 calls LOAD_ACCU procedure). Then the operation is performed by the OPERATION procedure. The intermediate result is stored in a buffer. The left operand, the operator, and the right operand are removed from the command string and replaced by the address of the intermediate buffer.
Line 22500 takes the string until the start of the first operand. Line 22502 takes the rest of the string after the right operator. Line 22504 takes from the original string the left and right parts that have not been analyzed during the current cycle. The address of the intermediate buffer is stored in h3$(e.g. 21). The new command string is assembled.

Case 1: Last operation has been performed on highest level parantheses

```
D  =  2  * ( 5.25 + 8.0 ) * 4
___________              _____

        lpo !  Z1      ! rpo
 h1b$          h3$        h5$
D  =  2  *  Z1   *  4
```

Case 2: Last operation has been performed on a command string with no parantheses

```
D    =    X2    +    X3    +    X4
_______                    _______

   h1b$  ! lpo ! opo ! rpo  h5$
         _____________
              Z1
              --
              h3$
D    =   Z1    +    X4
```

Case 3: Operation has been performed on all other command strings.

```
D   =   7   *  ( 8 + 4 * 3 + 1)  /  5
     ______

                _____ !  ___________
  h1h$          h2$   !  h4$    h5$
                      !
                      Z1
                     h3$
```

After the new command string has been assembled the interpreter procedure is run through again.

Operation procedure (lines 23000ff)
The operation procedure takes the contents of the left (X) accumulator and the right (Y) accumulator, performs the operation OPA and stores the result in an

intermediate buffer:
z(i) = REX(i) OP REY(.). Both operands can be signal values or one dimensional arrays. For arrays the operation is performed an%+1 times on the corresponding elements and the result elements are stored in the result array.

operator	code line	
0	23080	assignment z(i) = REX(i)
1	23100	exponentiation
2	23200	X log Y X = Basis
3	23300	multiplication
4	23400	division xi = 0 --> zi =0.00E20
5	23500	Differentiation
6	23600	mill
7	23700	+
8	23800	-

LOAD_ACCU procedure (lines 26000ff)
Depending on the data type the accumulators have to be loaded differently (see DATA_TYPE procedure for more information on data types). Data type 0 stands for no operands. In this case default values are taken for logarithm 2,71828182 and 1 for multiplication and division. For other operations default is zero. For data type 1 the ASCII string is converted into a real number and loaded an%+1 times into the accumulator (line 26070). For data type 2 the content of the intermediate buffer dx(lque,) is copied into the accumulator (line 26080). For data type 3 and 4 the X and Y arrays are read from the workfile (lines 26100 and 26110). For data type 5 the content of the constant c(lque) in copied an%+1 times into the accumulator (line 26120). For data type 6 the content of the header

variable lque2 from the workfile register lque is copied and%+1 times into the accumulator (line 26130). Data types A and B are used as destinations only (see STORE_RESULT procedure). For data types 9 and 10 one element out of the X or Y array of the register lque is copied an%+1 times into the accumulator (lines 26150 and 26160). The REY accumulator is loaded in a similar way using lines 26200 - 26297.

GET_DATA_TYPE procedure (lines 27000ff)
The GET_DATA_TYPE procedure searches for data type identifiers in the operand fields.

Datatype	identifier		
1	digits	9	real numbers
2	Z	Z1	intermediate buffer
3	X	X7	one dimensional array of X elements
4	Y	Y8	one dimensional array of Y elements
5	C	C1	constant
6	V	V1 1	variable (from header)
7	A	A1	output X array
8	B	B1	output Y array
9	N	N1 1	one element out of X array
10	M	M1 1	one element out of Y array

STORE_RESULT procedure (lines 25000ff)
The result destination has to be given to the left of the equation sign. Line 25510 analyses the right hand side of the command string. On line 25540 the data is loaded into the Y-accumulator. Next the left hand side

of the equation sign is analyzed and the data type is determined (lines 25550 - 25580). In addition to the usual data types the data can be sent to the display 'D' or to the printer 'P'. Depending on the data type a different message is shown (lines 25630 - 25590 and the result is stored.

6.3 Transformation Interpreter

INTP_TRANSFORM procedure (lines 19500ff)

The string for the X-transformation (format$(25)) is copied into the interpreter string area (in$). On line 19510 the undefined sources "RR" are filled with the vlaue of the sourece register (str$(tque(j))). On lines 19520 and 19530 adjacent registers can be accessed. If a "P1" is found the register following to tque(j) is used; if a "M1" is found, the register preceding to tque(j) is used. The string is then anded over to the mathematic interpreter. The string gor the Y-transformation (format$(26)) is processed in the same way before the procedure ends.

The signal for the interpreter to store the result in the X-output buffer (dx(n,)) is the character "A" followed by an integer number (1-10). If a "B" is found to the left lof the equation sign, the result is stored in the Y-output buffer (dy(n,)).

6.4 Equation Interpreter

Novel to direct curve fit programs on personal computers is a feature that is based on the operation interpreter described above.

The intepreter uses the equation entered with the function 'model'. To the left of the equality sign the dummy argument 'R' has to be entered. This argument channels the result of the model to the sum of squares routine. The right side has got tho contain at least once the substrate or inhibitor concentration (S).

During the fit, the values for the (S) symbol are read from the source record with the experimental data. The model may use up to 6 parameters (C1 to C6) that are optimized, so that the sum of the squares, of the differences between the model and the experimental data become minimum. Besides these entries, the model may contain any operators, constants, and other source records. Before the fit is executed, the number of data points of the experimental data is compared to the number of data points of the record containing the noise. If they are not equal, no fit is performable. Once the equation has been entered, the interpreter fit (model 10) is treated like other models. A source record, a noise record, and a destination record for the simulated curve have to be selected. The first estimates together with the finest grid factors are entered. The fit is started with the 'execute fit' function in the Fit Menu. Sample outputs of the fit routine are listed.
The equation interpreter works in a similar way as the tranformation interpreter. Two procedures are needed to connect the fit routine with the standard mathematic interpreter:

CONV_ADDR procedure (lines 18800ff)
This procedure reads the input string from the system file (format$(26)). As in the transformation interpreter the relative addresses "RR" are converted into absolute addresses using the pointer zr%.
INTP_FIT procedure (lines 19000ff)
The equation interpreter in the fit overlay uses the mathematic interpreter to evaluate fit models and to perform fits. The parameters of the fit (p1 to p6) are stored in the variables (C(1) to C(6)) of the

interpreter. The X-vector of the experimental data is accessed with S (substrate). The sum of squares is calculated if the character 'R' is found to the left of the equation sign. That means, the interpreter is called with the parameters p1 to p6 and to pointers to the experimental data and to the weights. It then returns the sum of squares of differences between the calculated signal and the experimental signal.
In the fit routine the interpreter is used to simulate a signal using an equation and an X-vector.
The command code for the interpreter to perform a sum of square is a character "R" on the left side of the equation sign.

Chapter 7
Data Structures

7.1 Common data Area

The common data block of the MICHFIT software is a reserved portion in RAM. It is used to exchange global variables between the subroutines. The global variables are explained in the table below:
Pointers and flags

pinit	initiation performed; this flag is set if the common data area has been initialized
forts	proceed pointer; this pointer is used to activate either input, content or output menu when the data processing subroutine is activated
tmod	transformation 1-6
tein	input device 1-6

taus	output device 1-6
teart	file type
idat	
name	name of user (used to identify workfile)
initialen	initials of user (used for file names)
pret	calling program is reactivated after termination of active subroutine
datstr	date generated by the main program is printed to curve fits and is shown by output menu
anzgem	No. to average
anztit	No. of curves in input and output menu
autoscale	austoscale on/off 1/0
interpolate	interpolate flag
anzeige	actual page in content menu
softk	active set of softkeys
offo%	roll state of output menu
offi%	roll state of input menu
format ()	prompts read from system files during start up sym () symbols read from system files during start up and when other symbols are needed
inhalt ()	directory contains file names of 32 files of the same file type (e.g. kinetic files)
inhp ()	pointer for number of entries of each file type in each drive (5 file types 5 drives)
macro ()	macros for command language

file name ()	file names are used to hand over file names to the input subroutine
iu ()	screen update pointers; is used to store the parts of the menus that have to be updated if the computer is not busy
ip ()	workfile update; is used to store the records that have been changed by external routines.
10 data curves	
psym(10)	symbols resp. line types (1-50)
stift%(10)	pen (0-8) for plotter and display
tque(20)	source registers in input and output menu
dmax(10)	pointers for the number of data points of each data curve in RAM
dx(10,332)	array of data curves
dy(10,332)	
definition of plot	
beschr ()	annotate strings, title, and labels of axes
tl(30)	string length
pg(20),pga(30	plot definitions
pxu#(),pxo#()	minimum, maximum, delta x and delta xo
pyl#(),pyr#()	minimum, maximum, delta y and delta yo
pgx(),pgy()	sizes of title, labels, digits, symbols and annotate
idx(),idy()	draw max. 128 data points
idp()	pen, symbol, or line type

touchfields

icon ()   definitions of touchfields

Chapter 8
List of Procedures

ALPHABETICAL LIST OF PROCEDURES

a) Data Handling Overlay

ADD_ENTRY PROCEDURE	1975ff
ADD_MUL PROCEDURE	8200ff
BACK_OUTPUT PROCEDURE	4300ff
CHECK_INPUT PROCEDURE	5000ff
DAHA_COMMAND PROCEDURE	3400ff
DAHA_MENU PROCEDURE	3200ff
DAHA_UPDATE PROCEDURE	3025ff
DATA_READ PROCEDURE	9200ff
DATA_WRITE PROCEDURE	9100ff
DEL_ENTRY PROCEDURE	1950ff
EDIT_CMD PROCEDURE	
EMERGENCY_CLEAN PROCEDURE	6400ff
ERROR_RECOVER PROCEDURE	6000ff
GET_DIRECTORY PROCEDURE	4200ff
GET_STRING PROCEDURE	40000ff
HEADER_READ PROCEDURE	9800ff
HEADER_WRITE PROCEDURE	9900ff
HELP PROCEDURE	6500ff
INCONVERT PROCEDURE	
INPUT_COMMAND PROCEDURE	1400ff
INPUT_FILENAME PROCEDURE	1900ff
INPUT_MENU PROCEDURE	1000ff
INPUT_UPDATE PROCEDURE	1025ff
KEYLABEL PROCEDURE	8700ff
OUTPUT_CMD PROCEDURE	2400ff
OUTPUT_MENU PROCEDURE	2000ff

OUTPUT_UPDATE PROCEDURE	2025ff
QUICK_HELP PROCEDURE	6900ff
READ_DEFAULT PROCEDURE	18000ff
READ_PLOTFILE PROCEDURE	7500ff
READ_SYSFILE PROCEDURE	18500ff
RETEXT PROCEDURE	4000ff
SET_TOUCH PROCEDURE	8500ff
SMOOTH PROCEDURE	7000ff
SPOOLER PROCEDURE	4350ff
START PROCEDURE	4100ff
STORE_MENU PROCEDURE	6800ff
TERMINATE PROCEDURE	850ff
TRANSFORM PROCEDURE	2900ff
TRANSFORM2 PROCEDURE	2905ff
WARN PROCEDURE	6300ff
WRITE_PF1 PROCEDURE	8035ff
WRITE_PLOTFILE PROCEDURE	8000ff
b) Input Overlay	
COPY_HEADER PROCEDURE	11500ff
DATA_READ PROCEDURE	9200ff
DATA_WRITE PROCEDURE	9100ff
HEADER_CHECK PROCEDURE	11300ff
HEADER_IN PROCEDURE	11000ff
HEADER_READ PROCEDURE	9800ff
HEADER_WRITE PROCEDURE	9900ff
IN_SERIAL PROCEDURE	12000ff
KEYBOARD_INPUT PROCEDURE	1900ff
READ_AVERAGE PROCEDURE	10000ff
READ_XYDATA PROCEDURE	19000ff
SWAP_WORKFILE PROCEDURE	3000ff
c) Interpreter	
GET_DATATYPE PROCEDURE	27000ff
LOAD_ACCU PROCEDURE	26000ff

OPERATION PROCEDURE	23000ff
STORE_RESULT PROCEDURE	25000ff
d) Fit Overlay	
CALCULATE PROCEDURE	6500ff
CHANGE_PAR PROCEDURE	7000ff
CHECKV PROCEDURE	5500ff
CONV_ADDR PROCEDURE	18800ff
EVALUATE PROCEDURE	5800ff
EXPAND PROCEDURE	7400ff
FIT PROCEDURE	5200ff
FITC PROCEDURE	6700ff
FIT_CMD PROCEDURE	7800ff
FIT_COMMAND PROCEDURE	3400ff
FIT_DEBUG PROCEDURE	12000ff
FIT_UPDATE PROCEDURE	12500ff
FIT_UPDATE2 PROCEDURE	12700ff
INPUT_FIT PROCEDURE	5000ff
INPUT_3NUM PROCEDURE	7100ff
INTERPRETER PROCEDURE	20000ff
INTP_FIT PROCEDURE	19000ff
INTP_SIMU PROCEDURE	18000ff
KEY_INPUT PROCEDURE	11000ff
LIN_REG PROCEDURE	10000ff
MODELS PROCEDURE	4900ff
PRINT_RESULT PROCEDURE	6900ff
PRINTER_OFF PROCEDURE	7700ff
RECOVER PROCEDURE	4200ff
SET_LIMIT PROCEDURE	5600ff
SIMULATION PROCEDURE	6000ff
SIMULATION2 PROCEDURE	6150ff
TIMEDATE PROCEDURE	7600ff
WEIGHT PROCEDURE	7500ff
e) Plot Overlay	

AUTOSCALE PROCEDURE	27000ff
BACK_OUT PROCEDURE	4300ff
CHAR_SIZE PROCEDURE	29100ff
CHECK_AXES PROCEDURE	23600ff
CHECK_INPUT PROCEDURE	5000ff
CLOSE_SPOOLFILE PROCEDURE	4400ff
CURSOR PROCEDURE	28000ff
DCURSOR PROCEDURE	28100ff
DCURVE PROCEDURE	23300ff
DISPLAY_ANNOTATE PROCEDURE	23090ff
DISPLAY_AXES PROCEDURE	23050ff
DISPLAY_CMD PROCEDURE	22400ff
DISPLAY_COLUMN PROCEDURE	23800ff
DISPLAY_DATA PROCEDURE	23250ff
DISPLAY_DRAW PROCEDURE	23700ff
DISPLAY_FRAME PROCEDURE	23100ff
DISPLAY_SCALE PROCEDURE	23000ff
DISPLAY_UPDATE PROCEDURE	22025ff
ERROR_RECOVER PROCEDURE	6000ff
FAST_CURSOR PROCEDURE	28140ff
FIND_OUTPUT PROCEDURE	7000ff
INPUT_2NUM PROCEDURE	3960ff
INTERPOLATE PROCEDURE	26500ff
LAYOUT_DEFAULTS PROCEDURE	29800ff
LINETYPE PROCEDURE	26100ff
KEY_INPUT PROCEDURE	11000ff
KEYLABEL PROCEDURE	8700ff
MOVE PROCEDURE	26000ff
PLOT_ANNOTATE PROCEDURE	25150ff
PLOT_AXES PROCEDURE	25192ff
PLOT_COLUMNS PROCEDURE	26800ff
PLOT_DATA PROCEDURE	25600ff
PLOT_DRAW PROCEDURE	25850ff
PLOT_FRAME PROCEDURE	23500ff

PLOT_LABEL PROCEDURE	25820ff
PLOTSIZE_CMD PROCEDURE	21400ff
PLOTSIZE_MENU PROCEDURE	21000ff
PLOTSIZE_UPDATE PROCEDURE	21025ff
POSITION_DEFAULTS PROCEDURE	29500ff
PRINT_DATA PROCEDURE	24080ff
PRINT_HEADER PROCEDURE	24000ff
READ_CHAR_SET PROCEDURE	26400ff
SENDPL PROCEDURE	25800ff
SET_TOUCH PROCEDURE	8500ff
SETUP_TOUCHSCREEN PROCEDURE	30500ff
SMOOTH PROCEDURE	27500ff
SYMBOL_MENU PROCEDURE	22673ff
XAXIS PROCEDURE	25900ff
XY_CURSOR PROCEDURE	28200ff
YAXIS PROCEDURE	25950ff

APPENDIX D
MICHKIN Software Listings

```
rem  ************************************************
rem  *****   MICHKIN software :  Autost.ksys    *****
rem  **********     Author : Bruno Michel         ****
rem  ************************************************
rem
rem  KINETIK B.MICHEL 02. Feb. 88
rem  STORE "Autost.KSYS"
rem
rem  ************ data structure *********************
rem  a) common data area
rem
rem  S1(80)         time course 1 (absorption vs. time
                    data and derivative)
rem  S2(80)         time course 2      do.
rem  S3(80)         average time course of several
                    measurements
rem  S4(10)         buffer area for up to 10
                    measurements (Delata AU per s)
                    rem  T1(41)titration curve 1 (Delta
                    AU per second)
rem  T2(41)         titration curve 2 (Delta AU per
                    second)
rem  D1$[1],D2 *    series of measurement (A-Z), and
                    number (0-999)
rem  D4$[20]   *    flags
rem  E1$[20]   *    name of enzyme
rem  E2$[20]   *    name of substrate
rem  T1$[64]   *    comment
rem  T(20)     *    measurement parameters
rem  L(20)     *    additional measurement parameters
rem  B(20)     *    additional measurement parameters
rem  W(20)      *    ASSAYOMATE syringe setup (volumes,
                               speed, filllevel)
rem  W1(20)     *    ASSAYOMATE stopped flow setup
rem  A(5,5)     *    axes of plots
rem  D3$[20]         date of measurement
rem  C(15)           counters of syringe volumes
rem  E(20)           enzyme concentration of probes in
                     job queue
rem  Z5-Z9           flags
rem  B9$[8]          name of volume (disc)
rem
rem       *          stored in measurement parameter
file (header)
rem
rem  ************************************************
rem  Measurement parameter file (header as above)
rem
rem  T(1)      start wavelength of range 1
rem  T(2)      stop wavelength of range 1
rem  T(3)      enzyme concentration
rem  T(4)      substrate concentration
rem  T(5)      number of data points
rem  T(6)      start time of time window at max.
               substrate conc.
rem  T(7)      stop time    do.
rem  T(8)      number of repetitons (2-10)
rem  T(9)      data format and type of model
rem  T(10)     start wavelength for internal reference
rem  T(11)     stop wavelength for internal reference
rem  T(12)     start wavelength of range 2
rem  T(13)     stop wavelength of range 2
rem  T(14)     maximal substrate concentration in assay
rem  T(15)     extinction coefficient of substrate at
               range 1
```

```
rem  T(16)      extinction coefficient of substrate at
                range 2
rem  T(17)      extinction difference (product -
                substrate) at range 1
rem  T(18)      extinction difference (product -
                substrate) at range 2
rem  T(19)      spacing type (e.g 1=arithmetic
                2=geometric 3=mixed)
rem  T(20)      additional repetitions (0-9)
rem  **************************************************
rem  B(1)       maximal noise of average in %
rem  B(2)       maximal nonlinearity of assay in %
rem  B(3)       actual enzyme concentration in assay
rem  B(4)       preselect enzyme concentration for
                automatic operation
rem  B(5)       actual substrate concentration in assay
rem  B(6)       assay volume in ml
rem  B(7)
rem  B(8)
rem  B(9)
rem  B(10)
rem  B(11)      measurement interval (s)
rem  B(12)      integration time (s)
rem  B(13)      start time of time window at minimal
                substrate conc.
                rem  B(14)stope time of time window at
                minimal substrate conc.
rem  B(15)
rem  B(16)
rem  B(17)
rem  B(18)
rem  B(19)
rem  B(20)
rem  **************************************************
rem  X0 - X3         minimum, maximum, delta x and delta
                     x0 of active axis
rem  Y0 - Y3         minimum, maximum, delta y and delta
                     y0 of active axis
rem  L1,L2           active integration time and active
                     interval (s)
rem  X4 - X7
rem  Y4 - Y7
rem  E3$[30]         label of X-axis
rem  E4$[40]         label of Y-axis
rem  E5$[60]
rem  H1$[60]
rem
rem  **************************************************
rem                           start of source code
rem
*******************************************************
rem  ********** definitions ***************************
rem  *********** define common data area **************
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
130 COM T1$[65],E1$[20],E2$[20].D$[20],D1$[1],
     D2,D3$[20],D4$[20],B9$[8]
rem  *********** define global variables **************
200 SHORT Y0,Y1,Y2,Y3,X0,X1,X2,X3,L1,L2,X4,X5,
     X6,X7,Y4,Y5,Y6,Y7
210 DIM E3$[30],E4$[30],E5$[60]
220 DIM H1$[60]
rem ******* test if initialisation necessary **********
250 ON ERROR GOTO 260
255 IF C(8)=1 THEN 4000
rem  ************* initialisation *********************
```

```
260 CLEAR @ C(8)=1 @ D2=10 @ C(9)=0 @ C(6)=1 @ C(15)=0
     @ GOSUB 265 @ GOTO 295
265 DISP "*********************************"
270 DISP "***  MICHKIN software for the **" @
    DISP "***  the autom. determination **"
275 DISP "***  of kinetic constants     **"
280 DISP "***  B.Michel        Version   **" @
    DISP "***           02. Feb. 88     **"
290 DISP "*********************************" @ RETURN
295 FOR I=1 TO 41 @ T1(I)=0 @ T2(I)=0 @ NEXT I
297 FOR I=1 TO 20 @ T(I)=0 @ L(I)=0 @ B(I)=0 @ W(I)=0 @
              E(I)=0 @ NEXT I
300 ! *************** initialize W-array **************
305 W(1)=50 @ W(2)=5 @ W(3)=2.5 @ W(4)=100 @ W(5)=1
307 ! max. volumes
310 W(6)=50 @ W(7)=5 @ W(8)=2.5 @ W(9)=100 @ W(10)=1
312 ! max. fill level
315 W(11)=1 @ W(12)=.5 @ W(13)=.25 @ W(14)=1 @ W(15)=1
317 ! amount per keypress
320 W(16)=1 @ W(17)=.1 @ W(18)=.05 @ W(19)=1 @
     W(20)=.01
322 ! syringe moving speed in cm/min
325 W1(1)=.4 @ W1(2)=.05 @ W1(3)=.05 @ W1(4)=60000 @
     W1(5)=60
327 ! set up of stopped flow with default values
330 C(1)=0 @ C(2)=0 @ C(3)=0 @ C(4)=0 @ C(5)=0
332 ! counters of syringe volumes internal to MICHKIN
340 ! ********* initialize T-array ******************
342 T(1)=548 @ T(2)=552 @ T(3)=.1 @ T(4)=150 @ T(5)=20
     @ T(6)=4 @ T(7)=10
344 ! wavelengths 1 range 1, enzyme and substrate
     conc., No.     of points and time window 1
346 T(8)=3 @ T(9)=1 @ T(10)=580 @ T(11)=586 @ T(12)=416
     @              T(13)=420 @ T(14)=30
```

```
348 ! no. of rep., data format, wavelengths int.
    reference, wavelengths range 2, max. Substr. conc.
350 T(15)=27.6 @ T(16)=60 @ T(17)=-16.3 @ T(18)=-43
352 ! epsilon substrate range 1, epsilon substr. range
    2 and delta epsilon 1 and 2
354 T(19)=1 @ T(20)=1 @ T(0)=T(8)
356 ! type of spacing  /  additional repetitions
358 T1$="................" @ T1$=T1$&T1$&T1$&T1$
359 ! ************ initialize B-array *****************
360 B(1)=1 @ B(2)=1 @ B(3)=.5 @ B(4)=0 @ B(5)=.5
362 ! max. noise % / Nonlinearity % / [E] in assay /
    [E]auto   / [E] in assay presel.
364 B(6)=10 @ B(7)=.5
366 ! actual [S] / assay volume
368 B(11)=.5 @ B(12)=.5 @ B(13)=1 @ B(14)=4
370 ! 11 : interval and 12 : integration time, 13/14
    timewindow at [S] min.
376 ! L(1) to L(10)= measuring wavelenths
377 FOR I=1 TO 20 @ L(I)=1 @ NEXT I
378 E1$=".................." @ E2$="Cytochrome c"
380 D4$="KAON0000       0" @ D1$="A" @ D2=10
386 ! ************ initialize plot defaults **********
388 A(1,1)=0 @ A(1,2)=100 @ A(1,3)=20 @ A(1,4)=0 ! TN
    vs. Zeit
390 A(2,1)=-.01 @ A(2,2)=.001 @ A(2,3)=.001 @ A(2,4)=0
    ! Delta OD Plot
392 A(3,1)=-.1 @ A(3,2)=1.5 @ A(3,3)=.2 @ A(3,4)=.1 !
    OD Plot
398 ! *************************************************
400 DISP @ FLIP @ DISP "Please enter the date and the
    name of the diskette"
401 DISP "eg.  30.Okt. 86,Michel"
402 DISP "The two inputs have to be se-    parated by a
    comma";@ INPUT D3$,B9$@ FLIP
```

```
403 ON ERROR GOTO 410
404 GOSUB 2000 @ GOTO 3900
410 DISP "File ";D$;" not found" @ DISP "Please select
     :" @ ON ERROR GOTO 4000
412 DISP "1 = new read" @ DISP "2 = rename" @ DISP "3 =
     new Directory"
414 DISP "4 = initialising" @ DISP "5 = show Directory
     " @ DISP "9 = Typing error"
420 INPUT H2
422 IF H2<2 OR H2>5 THEN 400
424 IF H2=3 THEN GOSUB 2200 @ GOTO 3900
426 DISP "Please enter Drive (0/1/2/3/4) 0=:D700  1
     =:D701  4=:D710" @ INPUT H1
430 IF H2=4 THEN 1650
435 ON ERROR GOTO 450 @ H1$=":D700:D701:D702:D703:D710"
     @ A$=H1$[H1*5+1,H1*5+5]
440 IF H2=5 THEN CLEAR @ DISP "Press (+cont)" @ CAT A$
     @ PAUSE
442 IF H2=5 THEN 412
443 ON ERROR GOTO 400
444 IF H2=2 THEN VOLUME A$ IS B9$ @ GOTO 402
448 BEEP @ GOTO 400
450 IF ERRN=130 THEN DISP "Please check the Drive " @
     DISP A$ @ BEEP
455 DISP "****** Error *******" @ DISP "******
     ";ERRN;" ******" @ BEEP
460 GOTO 400
rem *********** parameter menu ***********************
rem
500 ON ERROR GOTO 9000 @ CLEAR
505 DISP "******* Main programm **********" @ DISP
     "***** Mess Parameter Menu 1 ****"
510 DISP E1$;TAB(20);T(3);"nM (E)" @ DISP E2$;TAB(20);
     T(4):"M (S)"
```

```
515 DISP "[S]max in Asssay";TAB(22);T(14);"uM"
520 DISP "Range  1 from :";T(1);" to";T(2);"nm" @ DISP
     "Range  2 from :";T(12);" to";T(13);"nm"
525 DISP "Int. ref. form:";T(10);" to";T(11);"nm" @
     DISP "Epsilon   Range 1  and Range 2"
530 DISP "Substrate:";TAB(12);T(15);TAB(18);T(16);"1/mM
     cm"
535 DISP "Product  :";TAB(12);T(15)+T(17);TAB(18);
     T(16)+T(18);"1/mM cm"
540 DISP "No. of Points";TAB(24);T(5) @ DISP "No. of
     Repetitons";TAB(24);T(8);T(20)
rem  ********* first set of softkeys *****************
550 ON KEY# 1,"Exit" GOTO 4000
552 ON KEY# 2,"Points" GOTO 840
554 ON KEY# 3,"Range" GOTO 760
556 ON KEY# 4,"Page2" GOTO 600
558 ON KEY# 5,"Conc." GOTO 880
560 ON KEY# 6,"NameE" GOTO 690
562 ON KEY# 7,"[S]max" GOTO 890
564 ON KEY# 8,"Epsilon" GOTO 700
rem *********** second page of parameter menu ********
566 KEY LABEL @ GOTO 550
600 CLEAR @ DISP "***** Mess Parameter Menu 2 ****"
605 DISP "Durations:"
610 DISP "at [S]max :  ";T(6);" to";T(7);"sec" @ DISP
     "at  [S] min:  ";B(13);" to";B(14);"sec"
615 DISP "Max Standarddev.Av";B(1);" EzMsg";B(2);"%" @
     DISP "Data form: ";T(9);"  ";
620 IF T(19)=1 THEN DISP "Geom. Row"
622 IF T(19)=2 THEN DISP "Arithm. Row"
624 IF T(19)=3 THEN DISP "Mixed Row"
626 IF T(19)=4 THEN DISP "Reciprocal Row"
630 DISP "Y-Axis TN vs. [S] Plot :" @ DISP A(1,1);"
     to";A(1,2);"d";A(1,3);"d0";A(1,4)
```

```
635 DISP "Y-Axis delta OD vs Time Plot:" @ DISP  A(2,1
);" to";A(2,2);"d";A(2,3);"d0";A(2,4)
640 DISP "Y-Axis OD vs Time Plot:" @ DISP A(3,1);"
    to";A(3,2);"d";A(3,3);"d0";A(3,4)
rem *********** second set of softkeys ****************
670 ON KEY# 1,"Row" GOTO 995
672 ON KEY# 2,"Stdev" GOTO 920
674 ON KEY# 3,"Durat" GOTO 740
676 ON KEY# 4,"Pagel" GOTO 500
678 ON KEY# 5,"TN Sca" GOTO 960
680 ON KEY# 6,"OD Sca" GOTO 940
682 ON KEY# 7,ÖD Sca" GOTO 980
rem ********* routines performing softkey actions *****
684 ON KEY# 8,"Dform" GOTO 720
686 KEY LABEL @ GOTO 670
690 CLEAR @ DISP "Enter name of enzyme" @ FLIP @ INPUT
    E1$
695 DISP "Enter name of substrate" @ INPUT E2$@ FLIP @
    GOTO 500
700 CLEAR @ DISP "Enter the extiction coefficient of
    substrate and product at"
702 DISP T(1);" to  ";T(2);" nm  in (1/mM cm)" @ INPUT
    T(15),H1@ T(17)=H1-T(15)
704 DISP "Enter the extiction coeffizient of substrate
    and product at"
706 DISP T(12);" to  ";T(13);" nm  in (1/mM cm)" @
    INPUT T(16),H1@ T(18)=H1-T(16) @ GOTO 500
720 IF T(9)=1 THEN T(9)=2 @ GOTO 600
725 IF T(9)=2 THEN T(9)=3 @ GOTO 600 ELSE T(9)=1 @ GOTO
    600
740 CLEAR @ DISP "Enter duration at [S]max.";@ INPUT
    T(6),T(7)
741 DISP "Enter duration at [S]min.";@ INPUT
    B(13),B(14)
```

```
742 DISP "Enter measur interval and inte- gration
     time";@ INPUT L2,L1
744 IF L1>L2 OR L1 MOD .1#0 OR L2 MOD .1#0 THEN GOTO
     742
745 IF B(13)>B(14) THEN GOTO 741
746 IF T(7)/L2>80 OR T(7) MOD L2#0 THEN GOTO 740
747 IF T(7)/L2>40 AND T(9)>1 THEN GOTO 740
750 IF B(14)>T(7) THEN GOTO 740
755 B(15)=B(13) @ B(16)=T(7) @ B(11)=L1 @ B(12)=L2 @
     GOSUB 5100 @ GOSUB 5200 @ GOTO 600
760 CLEAR @ DISP "Enter wavelength Range 1";@ INPUT
     T(1),T(2)@ GOSUB 4800
800 DISP "Enter wavelength range 2";@ INPUT
     T(12),T(13)@ GOSUB 4800
810 DISP "Enter Wavelength Range for int. reference";
820 INPUT T(10),T(11)@ GOSUB 4800 @ GOTO 500
840 CLEAR @ DISP "Enter Number of Points";@ INPUT T(5)
842 DISP "Enter number of repetitions and additional
     repetions at [S]min.";@ INPUT T(8),T(20)
845 IF T(5)*T(8)>100 OR T(8)>10 OR T(5)>40 OR T(5)<0 OR
     T(8)<2 THEN GOTO 840
850 GOTO 500
880 CLEAR @ DISP "Enter conc. of enzyme (in nM)   and
     substrate (in uM)";
885 INPUT T(3),T(4)@ GOTO 500
890 CLEAR @ DISP "Maximal substrate conzentration in
     Assay (in uM)";@ INPUT T(14)
900 IF T(14)<T(4)/15 OR T(14)>T(4)/4 THEN GOTO 890
910 GOSUB 5000 @ GOTO 500
920 CLEAR @ DISP "Enter the max.standarddeviation of
     the average in %";@ INPUT B(1)
925 DISP "Enter the max. nonlinearity of  single
     measurements in %";@ INPUT B(2)
```

```
930 IF B(1)<1 OR B(1)>100 OR B(2)<1 OR B(2)>100 THEN
    GOTO 920 ELSE GOTO 600
940 CLEAR @ DISP "Enter Y-min, Y-max, delta Y and delta
    Yo for OD Plot"
945 INPUT A(3,1),A(3,2),A(3,3),A(3,4)
947 IF A(3,2)<A(3,1) OR A(3,3)>A(3,2)-A(3,1) OR
    A(3,4)<0 THEN GOTO 940
950 GOSUB 5100 @ GOTO 600
960 CLEAR @ DISP "Enter Y-min, Y-max, delta Y and delta
    Yo for V Plot";
965 INPUT A(1,1),A(1,2),A(1,3),A(1,4)
967 IF A(1,2)<A(1,1) OR A(1,3)>A(1,2)-A(1,1) OR
    A(1,4)<0 THEN GOTO 960
970 GOSUB 5000 @ GOTO 600
980 CLEAR @ DISP "Enter Y-min, Y-max, delta Y and delta
    Yo forOD Plot";
985 INPUT A(2,1),A(2,2),A(2,3),A(2,4)
987 IF A(2,2)<A(2,1) OR A(2,3)>A(2,2)-A(2,1) OR
    A(2,4)<0 THEN GOTO 980
990 GOSUB 5200 @ GOTO 600
995 IF T(19)=1 THEN T(19)=2 @ GOTO 600
996 IF T(19)=2 THEN T(19)=3 @ GOTO 600
997 IF T(19)=3 THEN T(19)=4 @ GOTO 600 ELSE T(19)=1 @
    GOTO 600
rem ***************** setup menu **********************
rem
1000 ! ******** Syringe P******
1010 CLEAR @ DISP "**** Syringe Parameter Menu ****" @
     DISP
1030 DISP "     Volume  Fill   Mpk  ml/sec"
1040 DISP "Syr1:":TAB(8);W(1):TAB(14);W(6);TAB(20);
     W(11);TAB(26):W(16)
1045 DISP "Syr2:":TAB(8):W(2):TAB(14):W(7):TAB(20):
     W(12):TAB(26):W(17)
```

```
1050 DISP "Syr3:";TAB(8);W(3);TAB(14);W(8);TAB(20);
     W(13);TAB(26);W(18)
1055 DISP "Tray:";TAB(8);W(4);TAB(14);W(9);TAB(20);
     W(14);TAB(26);W(19)
1060 DISP "Need:";TAB(8);W(5);TAB(14);W(10);TAB(20);
     W(15);TAB(26);W(20)
1070 DISP "Stopfl.";W1(1);W1(2);W1(3)
1075 DISP "Accel.";W1(5);" Schrr ";W1(4)
1080 DISP "Volume per Assay";B(7);"ml"
rem *************** softkeys **************************
1150 ON KEY# 1,"Exit" GOTO 4000
1160 ON KEY# 2,"Stpfl" GOTO 1350
1170 ON KEY# 3,"Accel" GOTO 1300
1180 ON KEY# 4,"Needle" GOTO 1480
1190 ON KEY# 5,"Syr.1" GOTO 1400
1200 ON KEY# 6,"Syr.2" GOTO 1420
1210 ON KEY# 7,"Syr.3" GOTO 1440
1220 ON KEY# 8,"Tray" GOTO 1460
1230 KEY LABEL @ GOTO 1150
rem ********* routines performing softkey actions *****
1300 CLEAR @ DISP "Enter Acceleration for stoppped
     flow:";W1(5);" neu";@ INPUT W1(5)
1310 CLEAR @ DISP "Enter No. of cycles for
     stopppedFlow:";W1(4);" neu";@ INPUT W1(4)
1320 GOTO 1000
1350 CLEAR @ DISP "Enter the amounts for stopped  Flow
     former :";W1(1);W1(2);W1(3);"ml"
1355 DISP " new ";@ INPUT W1(1),W1(2),W1(3)@
     B(7)=W1(1)+W1(2)+W1(3)
1360 GOTO 1000
1400 CLEAR @ DISP "Enter parameter for syringe
     1:",W(1); W(6);W(11);W(16);" new";
1410 INPUT W(1),W(6),W(11),W(16)@ GOTO 1000
```

```
1420 CLEAR @ DISP "Enter parameter for syringe 2",W(2);
     W(7);W(12);W(17);" new";
1430 INPUT W(2),W(7),W(12),W(17)@ GOTO 1000
1440 CLEAR @ DISP "Enter parameter forsyringe 3:",W(3);
     W(8);W(13);W(18);" new";
1450 INPUT W(3),W(8),W(13),W(18)@ GOTO 1000
1460 CLEAR @ DISP "Enter parameters for Tray  :",W(4);
     W(9);W(14);W(19);" new";
1470 INPUT W(4),W(9),W(14),W(19)@ GOTO 1000
1480 CLEAR @ DISP "Enter parameters for needle
     :",W(5);W(10);W(15);W(20);" new";
1490 INPUT W(5),W(10),W(15),W(20)@ GOTO 1000
1600 CLEAR @ DISP "Enter drive: 0=':D700' 1=':D701' and
     4=:D710'";@ INPUT H1
1602 IF H1=0 THEN MASS STORAGE IS ":D700" @ GOTO 1605
1604 IF H1=1 THEN MASS STORAGE IS ":D701" ELSE MASS
     STORAGE IS ":D710"
1605 CAT @ DISP "Enter filename to erase or
     (PACK/INIT)"
1610 INPUT H1$
1620 IF H1$="PACK" THEN PACK @ GOTO 4000
1625 IF H1$="INIT" THEN GOTO 1650
1630 IF H1$="N" THEN GOTO 4000
1635 IF H1$="" THEN GOTO 4000
1640 PURGE H1$ @ GOTO 1610
1650 IF H1=0 THEN DISP "Systemdisk must not be
     initialized" @ WAIT 2000 @ GOTO 4000
1657 DISP "Are you sure to erase all data  on this
     diskette (J/N)";@ INPUT H1$
1658 IF H1$#"J" THEN 4000
1660 DISP "Enter new name of diskette" @ INPUT H2$
1665 DISP "Please wait about 2 minutes"
1670 H1$=":D700:D701:D702:D703:D710" @
     A$=H1$[H1*5+1,H1*5+5]
```

```
1675 INITIALIZE H2$,A$,20
1690 GOSUB 2200 @ GOTO 4000
rem ***************** utilities ***********************
2000 CLEAR @ DISP "The active Data diskette is  :  " @
     DISP "*****   ";B9$;"   *****"
2010 DISP "It contains data about   :   "
2025 D$="Inhalt."&B9$
2030 ASSIGN# 1 TO D$
2040 ON ERROR GOTO 2080
2050 READ# 1 ; H1$@ DISP H1$
2060 GOTO 2050
2080 ASSIGN# 1 TO *
2090 RETURN
2200 D$="Inhalt."&B9$ @ ON ERROR GOTO 2205 @ PURGE D$
2205 ON ERROR GOTO 9000 @ CREATE D$,2,256
2210 ASSIGN# 1 TO D$
2220 ON ERROR GOTO 2290
2230 CLEAR @ DISP "Please enter max . 12 lines of
     comment " @ I=1
2250 DISP "Zeile:";I @ FLIP @ INPUT H1$@ FLIP
2255 IF LEN(H1$)=0 OR LEN(H1$)>32 OR I>12 THEN 2290
2260 PRINT# 1 ; H1$ @ I=I+1 @ GOTO 2250
2290 ON ERROR GOTO 9000
2295 ASSIGN# 1 TO *
2390 RETURN
2395 ON ERROR GOTO 4000
3900 D$="DEFAULT.KSYS"
3905 ON ERROR GOTO 4000
3910 GOTO 4410
rem ***************** main menu ***********************
rem
4000 ! ***MAIN MENU***
4020 CLEAR @ GOSUB 265
4100 DISP "Please select a softkey:"
```

```
4110 DISP "all other keys stop the program"
4130 ON ERROR GOTO 9000
rem ****************** softkeys ***********************
4150 ON KEY# 1,"Km Msg" GOTO 4250
4160 ON KEY# 2,"Dacom" GOTO 4320
4170 ON KEY# 3,"ReadV" GOTO 4400
4180 ON KEY# 4,"MeasPa" GOTO 500
4190 ON KEY# 5,"Direct" GOTO 1600
4200 ON KEY# 7,"Sto Va" GOTO 4500
4210 ON KEY# 6,"Test M" GOTO 4340
4220 ON KEY# 8,"SyrinPa" GOTO 1000
4230 KEY LABEL @ GOTO 4230
rem ******** routines performing softkey actions ******
4250 DISP "The Measureprogram is loaded    Please wait
     about 30 seconds" @ DISP @ BEEP
4270 GOSUB 4600 ! ASSIGN
4280 GOSUB 5000 @ GOSUB 5100 @ GOSUB 5200 ! GENERATE
     PLOTS
4290 ON ERROR GOTO 4295 @ CREATE "TEMP1.KSYS",10,768 @
     CREATE "TEMP.KSYS",40,768
4295 SECURE "TEMP1.KSYS","AA",3 @ SECURE
     "TEMP.KSYS","AA",3
4300 REM
4306 ON ERROR GOTO 4310 @ ERASE STATUS
4307 FOR I=1 TO 99 @ ERASE STANDARD I @ NEXT I
4310 ON ERROR GOTO 9000
4315 IF T(9)=1 THEN D2=2 @ E3$="KIN 2.KSYS" ELSE D2=1 @
     E3$="KIN 1.KSYS"
4317 IF C(15)=0 THEN CHAIN "ASINIT.KSYS" ELSE D2=0 @
     CHAIN E3$
4320 DISP "The Datatransferprogram is      loaded
     " @ DISP @ BEEP
4330 CHAIN "DACOM.KSYS"
4340 DISP "The Testmeasure programm is     loaded "
```

```
4345 DISP "Please wait about 20 seconds" @ DISP @ BEEP
4360 GOSUB 4600 ! ASSIGN
4370 GOSUB 5000 @ GOSUB 5100 @ GOSUB 5200 ! GENERATE
     PLOTS
4380 E3$="KONZMSG.KSYS" @ D2=3
4385 IF C(15)=0 THEN CHAIN "ASINIT.KSYS" ELSE D2=0 @
     CHAIN E3$
4400 CLEAR @ CAT ".KSYS"
4405 DISP "Please enter the Filename" @ INPUT D$@
     D$=D$&".KSYS"
4410 ASSIGN# 1 TO D$
4412 ON ERROR GOTO 4450
4420 READ# 1 ; D1$,H1,D4$,D4$.E1$.E2$,T1$,T(),L(),
     B(),A(,),W(),W1()
4430 ASSIGN# 1 TO * @ ON ERROR GOTO 9000
4440 GOSUB 5000 @ GOSUB 5100 @ GOSUB 5200
4445 GOTO 4000
4450 DISP "Old File : Please update Syringe Parameters"
4490 GOTO 4430
4500 CLEAR @ CAT ".KSYS"
4505 DISP "Please enter the Filename" @ INPUT D$
4507 D$=D$&".KSYS"
4510 GOSUB 4600 ! ASSIGN
4520 ON ERROR GOTO 4565
4560 CREATE D$,5,256
4565 ON ERROR GOTO 9000 @ ASSIGN# 1 TO D$
4570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B(),
     A(,),W(),W1()
4580 ASSIGN# 1 TO *
4590 GOTO 4000
rem ********* assign photodiodes *********************
4600 ! ****ASSIGN ***
4602 GOSUB 4800
4605 FOR I=1 TO 10 @ L(I)=178 @ NEXT I @ I1=0
```

```
4610 FOR I=T(1) TO T(2) STEP 2
4620 I1=I1+1 @ L(I1)=I
4630 NEXT I
4640 FOR I=T(12) TO T(13) STEP 2
4650 FOR J=1 TO 10 @ IF I=L(J) THEN GOTO 4665
4655 IF I1=10 THEN BEEP @ DISP "ZU GROSSER
     WELLENLAENGENBEREICH" @ GOTO 500
4660 I1=I1+1 @ L(I1)=I
4665 NEXT I
4670 FOR I=T(10) TO 820 STEP 2
4672 FOR J=1 TO 10 @ IF I=L(J) THEN GOTO 4680
4675 I1=I1+1 @ L(I1)=I
4677 IF I1=10 THEN GOTO 4690
4680 NEXT I
4690 H2$=" " @ REM
rem ********* check validity of parameters ************
4700 IF T(3)<.0001 OR T(3)>10000 OR T(4)<.001 OR
T(4)>100000    THEN H1=3 @ H2=4 @ GOSUB 4795
4705 IF T(5)<2 OR T(5)>40 OR T(8)<2 OR T(8)>10 OR
     T(5)*T(8)>200 THEN H1=5 @ H2=8 @ GOSUB 4795
4710 IF T(9)>3 OR T(9)<1 OR T(19)<1 OR T(19)>5 THEN
     H1=9 @ H2=19 @ GOSUB 4795
4715 IF T(14)>T(4)/5 OR T(14)<T(4)/100 THEN H1=14 @
     H2=0 @ GOSUB 4795
4720 IF T(6)<.2 OR T(6)>100 OR T(7)<T(6)+1 OR T(7)>2000
     THEN H1=6 @ H2=7 @ GOSUB 4795
4725 IF T(15)<.001 OR T(15)>1000 OR T(16)<.001 OR
     T(16)>1000 THEN H1=15 @ H2=16 @ GOSUB 4795
4730 IF T(17)<-100 OR T(17)>100 OR T(18)<-100 OR
     T(18)>100 THEN H1=17 @ H2=18 @ GOSUB 4795
4735 IF B(1)<1 OR B(1)>100 OR B(2)<1 OR B(2)>100 THEN
     H1=21 @ H2=22 @ GOSUB 4795
4750 IF B(7)<.1 OR B(7)>10 THEN H1=27 @ H2=28 @ GOSUB
     4795
```

```
4760 IF B(11)<.2 OR B(11)>20 OR B(12)<B(11) OR
     B(12)>100 THEN H1=31 @ H2=32 @ GOSUB 4795
4765 IF B(13)<.2 OR B(13)>B(14) OR B(14)>T(7) OR
     B(14)>1000 THEN H1=33 @ H2=7 @ GOSUB 4795
4780 IF H2$="F" THEN BEEP @ WAIT 2000 @ GOTO 4000
4790 RETURN
4795 IF H1<20 THEN H1$="T" ELSE H1=H1-20 @ H2=H2-20
4796 IF H1<20 THEN H1$="B" ELSE H1=H1-20 @ H2=H2-20 @
     H1$="W"
4797 IF H2$=" " THEN CLEAR @ H2$="F"
4798 BEEP @ DISP "Variables ";H1$;H1;" or ";H2;"are
     invalid"
4799 WAIT 2000 @ RETURN
rem ************** check diode assignments ************
4800 REM *** TEST**
4810 IF T(1)>820 OR T(1)<180 OR T(1) MOD 2#0 THEN
     T(1)=200 @ GOTO 4900
4815 IF T(2)>820 OR T(2)<180 OR T(2) MOD 2#0 THEN
     T(2)=200 @ GOTO 4900
4820 IF T(10)>820 OR T(10)<180 OR T(10) MOD 2#0 THEN
     T(10)=200 @ GOTO 4900
4825 IF T(11)>820 OR T(11)<180 OR T(11) MOD 2#0 THEN
     T(11)=200 @ GOTO 4900
4830 IF T(12)>820 OR T(12)<180 OR T(12) MOD 2#0 THEN
     T(12)=200 @ GOTO 4900
4835 IF T(13)>820 OR T(13)<180 OR T(13) MOD 2#0 THEN
     T(13)=200 @ GOTO 4900
4850 IF (T(2)-T(1)+T(11)-T(10)+T(13)-T(12))/2+3>10 THEN
     DISP "Zuviele Dioden " @ BEEP @ GOTO 500
4880 RETURN
4900 BEEP @ DISP "invalid Wavelengths" @ WAIT 2000 @
     GOTO 500
rem *********** prepeare axes for plots **************
5000 REM ****PLOT V VS [S]*****
```

```
5005 D$="PFTN.KSYS"
5010 Y0=A(1,1) @ Y1=A(1,2) @ Y2=A(1,3) @ Y3=A(1,4)
5020 X0=0 @ X1=T(14)*1.05 @ X2=.02 @ X3=0 @ IF T(14)>.2
     THEN X2=.1
5030 IF T(14)>.4 THEN X2=.2
5035 IF T(14)>2 THEN X2=1
5040 IF T(14)>4 THEN X2=2
5042 IF T(14)>20 THEN X2=5
5044 IF T(14)>50 THEN X2=10
5046 IF T(14)>100 THEN X2=50
5048 IF T(14)>500 THEN X2=500
5050 E3$=E2$&" (uM)"
5060 E4$="Turnovernumber (1/sec)"
5090 GOSUB 6000 @ RETURN
5100 REM *****PLOT OD VS T*****
5105 D$="PFOD.KSYS"
5110 Y0=A(3,1) @ Y1=A(3,2) @ Y2=A(3,3) @ Y3=A(3,4)
5120 X0=0 @ X1=T(7) @ X2=2 @ X3=0 @ IF T(7)>15 THEN
     X2=5
5130 IF T(7)>50 THEN X2=10
5140 E3$="Time (sec)"
5150 E4$="Absorption"
5190 GOSUB 6000 @ RETURN
5200 REM **PLOT Ableit.**
5205 D$="PFAB.KSYS"
5210 Y0=A(2,1) @ Y1=A(2,2) @ Y2=A(2,3) @ Y3=A(2,4)
5220 X0=0 @ X1=T(7) @ X2=2 @ X3=0 @ IF T(7)>15 THEN
     X2=5
5230 IF T(7)>50 THEN X2=10
5240 E3$="Time (sec)"
5250 E4$="Speed (Delta OD/sec)"
5290 GOSUB 6000 @ RETURN
rem  ************ store plotter code in file *********
6000 ! PLOTTEN VON ACHSEN
```

```
6005 ON ERROR GOTO 6020 @ ASSIGN# 1 TO D$ @ READ# 1 ;
     P3,P4,X4,X5,X6,X7,Y4,Y5,Y6,Y7
6006 ASSIGN# 1 TO *
6007 IF X0#X4 OR X1#X5 OR X2#X6 OR X3#X7 THEN GOTO 6020
6008 IF Y0#Y4 OR Y1#Y5 OR Y2#Y6 OR Y3#Y7 THEN GOTO 6020
6010 RETURN
6020 ON ERROR GOTO 4000
6030 P3=2*400 @ P4=1*400 @ P5=(2+25)*400 @
     P6=(1+16)*400
6033 H1$=CHR$(3) @ P1=.000001 @ P2=.0000001
6036 F5=X1-X0 @ F6=Y1-Y0
6040 IF ABS(F5*P1)<3000 THEN P1=P1*10 @ GOTO 6040
6050 IF ABS(F6*P2)<3000 THEN P2=P2*10 @ GOTO 6050
6060 F5=(P5-P3)/100 @ F6=(P6-P4)/100
6070 ON ERROR GOTO 6090
6080 CREATE D$,1,5120
6090 ON ERROR GOTO 4000
6100 ASSIGN# 1 TO D$
6105 PRINT# 1 ; P1,P2,X0,X1,X2,X3,Y0,Y1,Y2,Y3,E3$,E4$
6110 E5$="IN;SP1;" @ GOSUB 6400
6120 E5$="IP"&VAL$(P3)&","&VAL$(P4)&","&VAL$(P5)&",
     "&VAL$(P6)&";" @ GOSUB 6400
6130 E5$="SC 0,10000,0,10000;" @ GOSUB 6400
6140 E5$="PA4000,9500;" @ GOSUB 6400 @ E5$="DT"&H1$&";"
     @ GOSUB 6400
6170 E5$="DI 1,0;" @ GOSUB 6400
6180 E5$="PU;PA 1300,1300;PD;PA 1300,9300;" @ GOSUB
     6400
6190 E5$="PA 9300,9300,9300,1300,1300,1300:PU;" @ GOSUB
     6400
6200 FOR I1=X0+X3 TO X1 STEP X2 @ I2=INT((I1-
X0)*8000/(X1-X0)+1300)
6210 E5$="PU;PA"&VAL$(I2)&",1300;PD:XT:PU;" @ GOSUB
     6400
```

```
6220 E5$="SI.2,.25;CP-1,-1.2;LB"&VAL$(I1)&H1$&";" @
     GOSUB 6400 @ NEXT I1
6230 FOR I1=Y0+Y3 TO Y1 STEP Y2 @ I2=INT((I1-
     Y0)*8000/(Y1-Y0)+1300)
6240 E5$="PA9300,"&VAL$(I2)&";PD;YT;PU;" @ GOSUB 6400 @
     NEXT I1
6250 FOR I1=Y0+Y3 TO Y1 STEP Y2 @ I2=INT((I1-
     Y0)*8000/(Y1-Y0)+1300)
6260 E5$="PA1300,"&VAL$(I2)&";PD;YT;PU;" @ GOSUB 6400
6270 E5$="CP-5,-.5;LB"&VAL$(I1)&H1$&";" @ GOSUB 6400 @
     NEXT I1
6280 FOR I1=X0+X3 TO X1 STEP X2 @ I2=INT((I1-
     X0)*8000/(X1-X0)+1300)
6290 E5$="PU;PA"&VAL$(I2)&",9300;PD;XT;PU;" @ GOSUB
     6400 @ NEXT I1
6300 P3=INT(P3+F5*13)-50 @ P4=INT(P4+F6*13) @
     P5=INT(P5-F5*7)-470 @ P6=INT(P6-F6*7)
6310 E5$="IP"&VAL$(P3)&","&VAL$(P4)&","&VAL$(P5)&",
     "&VAL$(P6)&";" @ GOSUB 6400
6320 E5$="IW"&VAL$(P3)&","&VAL$(P4)&","&VAL$(P5)&",
     "&VAL$(P6)&";" @ GOSUB 6400
6330 H1=INT(X0*P1) @ H2=INT(X1*P1) @ H3=INT(Y0*P2) @
     H4=INT(Y1*P2)
6340 E5$="SC"&VAL$(H1)&","&VAL$(H2)&","&VAL$(H3)&",
     "&VAL$(H4)&";" @ GOSUB 6400
6350 ASSIGN# 1 TO *
6360 SECURE D$,"XX",3 @ RETURN
6390 !  TO DISK **
6400 PRINT# 1 ; CHR$(34),E5$,CHR$(34) @ RETURN
rem ************* error recovery ********************
9000 ! **error recovery ***
9010 BEEP @ BEEP @ BEEP
9100 IF ERRN=1 THEN DISP "too small number" @ GOTO 9500
9110 IF ERRN=2 THEN DISP "too large number" @ GOTO 9500
```

```
9120 IF ERRN=7 THEN DISP "Null Data" @ GOTO 9500
9130 IF ERRN=8 THEN DISP "Division by zero" @ GOTO 9500
9140 IF ERRN=11 THEN DISP "Argument out of Range" @
     GOTO 9500
9150 IF ERRN=22 THEN DISP "File is secured" @ GOTO 9500
9155 IF ERRN=43 THEN DISP "Numeric input " @ GOTO 9500
9160 IF ERRN=44 THEN DISP "not enough inputs" @ GOTO
     9500
9170 IF ERRN=45 THEN DISP "too many inputs" @ GOTO 9500
9180 IF ERRN=49 THEN DISP "Null Data" @ GOTO 9500
9190 IF ERRN=56 THEN DISP "String Overflow" @ GOTO 9500
9200 IF ERRN=55 THEN DISP "String Subscript" @ GOTO
9500
9210 IF ERRN=67 THEN DISP "Filename doesn't exist" @
     GOTO 9500
9215 IF ERRN=68 THEN DISP "Filetyp wrong" @ GOTO 9500
9220 IF ERRN=89 THEN DISP "invalid parameter" @ GOTO
     9500
9230 IF ERRN=108 THEN DISP "Photometer Waring" @ GOTO
     9500
9240 IF ERRN=111 THEN DISP "No disk inserted" @ GOTO
     9500
9250 IF ERRN=125 THEN DISP "Volume not found" @ GOTO
     9500
9260 IF ERRN=128 THEN DISP "Disk is full" @ GOTO 9500
9265 IF ERRN=130 THEN DISP "no disk" @ GOTO 9500
9270 IF ERRN=131 THEN DISP "Time Out" @ GOTO 9500
9490 DISP "Error ";ERRN;" ocurred on line ";ERRL @
     PAUSE
9495 GOTO 4000
9500 WAIT 3000 @ GOTO 4000

rem  ***************************************************
rem  **********   MICHKIN software :  kin 2.KSYS    ***
rem  **********   Author : Bruno Michel          ***
rem  **************************************************
rem
rem
rem **************************************************
rem                  start of source code
rem **************************************************
rem *********1 ! KIN 2 B.MICHEL 02.02.88 *************
rem *************** common data area ******************
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
     D3$[20],D4$[20],B9$[8]
rem ************** global variables ******************
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],F5$[1],
     A$[60],H1$[20]
195 INTEGER I,I2,I5,I6
200 F6$="0" @ Q1=1 @ Q2=1 @ F7$="0" @ CLEAR @ I=0
210 F$="A" @ F5$="0" @ J=1 @ K=1 @ D4$[5,5]="1"
rem *********** chain Autost if not initialized *******
250 ON ERROR GOTO 800 @ PLOTTER 1 @ IF C(8)#1 THEN GOTO
     800
rem ************ measure menu ************************
rem
500 L1=B(11) @ L2=B(12) @ L3=256^(1/T(5)) @ ON ERROR
     GOTO 500
510 CLEAR @ DISP "    Km Measure : Main Menu" @ DISP
520 DISP E1$;TAB(19);T(3);"nm (E)" @ DISP E2$;TAB(19);
     T(4);"uM (S)" @ DISP
530 DISP "No. of Rep.";T(8) @ DISP "No. of Points
     ";T(5)
rem ********** first set of softkeys ****************
```

```
650 ON KEY# 1,"DiVar" GOTO 720 @ ON KEY# 2,"MeasM" GOTO
     1000 @ ON KEY# 3,"Syringe" GOTO 860
655 ON KEY# 4,"Wash" GOTO 700 @ ON KEY# 5,"MAINPG" GOTO
     800 @ ON KEY# 6,"TestM" GOTO 790
660 ON KEY# 7,"[E,S]" GOTO 880 @ ON KEY# 8,"Points"
     GOTO 940 @ KEY LABEL @ GOTO 650
rem ** routines performing actions of 1st set of
     softkeys **
700 GOSUB 7500 @ GOTO 500
720 PRINTER IS 1 @ GOSUB 9545 @ PAUSE
725 GOTO 500
760 IF B(4)=0 THEN B(4)=1 ELSE B(4)=0 @ CLEAR @ DISP
     "Enter [E] in Assay";@ INPUT B(5)
765 B(3)=B(5) @ GOTO 1000
790 RESET 7 @ CHAIN "KONZMSG.KSYS"
800 RESET 7 @ CHAIN "Autost.KSYS"
820 CLEAR @ DISP "** Single Meas **"
825 DISP E2$ @ DISP "Conc in uM at [S]<0: Reference";
830 INPUT B(6)@ C(10)=1
835 DISP E1$ @ DISP "Conc in nM (Stock=";T(3);"nM)";@
     INPUT B(5)
840 B(3)=B(5) @ MODE 0,1 @ GOSUB 4800 @ GOTO 1000
860 GOSUB 7250 @ GOTO 500
880 CLEAR @ DISP E1$;" Conc. in (nM)"
885 DISP "and ";E2$;" Conc. in (uM)" @ INPUT T(3),T(4)
890 GOTO 500
900 CLEAR @ IF B(20)=2 THEN DISP E1$:ELSE DISP E2$;
905 DISP " Conc. Nr.";C(6)
910 DISP "in Syringe (C=clear Buffer)" @ DISP "negative
     [E] = decreasing [S]" @ INPUT H1$
915 IF H1$="C" THEN C(6)=1 @ GOTO 900
920 E(C(6))=VAL(H1$) @ C(6)=C(6)+1 @ RETURN
940 CLEAR @ DISP "No. of Points (2-40) and Rep. (2-
     10)":
```

```
950 INPUT T(5),T(8)@ IF T(5)>40 OR T(5)<2 OR T(8)>10 OR
     T(8)<0 THEN GOTO 940 ELSE GOTO 500
rem ********* measure menu second part ****************
rem
1000 CLEAR @ DISP "    Km Measure Menu" @ DISP @ DISP
     T1$[1,63] @ DISP @ ON ERROR GOTO 500
1020 DISP "Syr.1:";INT(C(1)*10)/10;"2:";INT(C(2)*100)
     /100;"3:";INT(C(3)*100)/100;"ml" @ DISP
1050 DISP "Max. [S]  :  ";T(14);"uM"
1090 IF D4$[4,4]="L" THEN DISP "Blank ";D4$[6,14];"
     subtracted" ELSE DISP "No Blank"
1120 IF B(4)=1 THEN DISP "[E] Autom."
1130 IF D4$[5,5]="1" THEN DISP "Print On" ELSE DISP
     "Print OFF"
1140 IF C(9)=0 THEN DISP "No Reference" @ BEEP
1150 DISP "Assay Volume : ";B(7);"ul" @ ON ERROR GOTO
1000
rem ***************** softkeys ************************
1200 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"Print" GOTO
     1280 @ ON KEY# 3,"Content" GOTO 1600
1230 ON KEY# 4,"Comment" GOTO 1690 @ ON KEY# 5,"KmMeas"
     GOTO 1300 @ ON KEY# 6,"Blank" GOTO 1500
1260 ON KEY# 7,"[E]auto" GOTO 760 @ ON KEY#
     8,"SingleMe" GOTO 820 @ KEY LABEL @ GOTO 1200
rem *** routines performing softkey actions 2nd set ***
1280 IF D4$[5,5]="0" THEN D4$[5,5]="1" ELSE
     D4$[5,5]="0"
1290 GOTO 1000
rem *** start determination of kinetic constant *******
1300 ON ERROR GOTO 1000 @ IF C(9)=0 THEN GOTO 1000
1302 CLEAR @ DISP "Series and Number eg A,1";@ INPUT
     D1$,B1@ IF B1>999 OR B1<0 THEN 1300
1304 DISP "Memocode (6 Characters)";@ INPUT D4$[15,20]
1310 M=B(20) @ IF C(M)<W(M)/8 THEN H1=7 @ GOSUB 7510
```

```
1320 D2=B1 @ B1=B1+1 @ F8$="N"
1325 IF C(6)=1 THEN GOSUB 900 ELSE H1=0 @ PRINTER IS 1
     @ GOSUB 9500
1330 B(3)=B(5) @ IF E(1)#0 THEN Q1=1 @ GOTO 1335
1332 E(1)=1 @ B(3)=.1 @ Q1=2 @ D4$[4,4]="0" @
     D1$=CHR$(NUM(D1$)+1) @ D2=0
1335 IF E(1)<0 THEN Q2=2 @ E(1)=-E(1) ELSE Q2=1
1337 IF B(20)=2 THEN T(3)=E(1) ELSE T(4)=E(1)
1340 FOR I=1 TO C(6)-1 @ E(I)=E(I+1) @ NEXT I @
     C(6)=C(6)-1
1350 W1(2)=B(7)*B(5)/T(3) @ GOSUB 8000 @ GOTO 2000
1380 IF C(6)<2 THEN 1000 ELSE CLEAR @ GOTO 1320
rem  ************* activate blank subtraction *********
1500 IF D4$[4,4]="L" THEN D4$[4,4]="0" @ GOTO 1000
1510 CLEAR @ DISP "Name of Blankfile (9 Characters)";@
     INPUT H1$@ D4$[4,4]="L"
1515 IF LEN(H1$)<9 THEN GOTO 1510
1520 H1$=H1$&"    " @ D4$[6,14]=H1$
1540 CREATE D4$[6,14]&".KSYS",5,256 @ ASSIGN# 1 TO
     D4$[6,14]&".KSYS"
1560 FOR J=1 TO 20 @ PRINT# 1 ; T(J) @ NEXT J
1570 FOR J=0 TO T(5) @ PRINT# 1 ; T1(J),T2(J) @ NEXT J
1580 ASSIGN# 1 TO * @ H1$=D4$[6,14]
1590 PRINTER IS 2 @ PRINT "Blank Stored in File: ";H1$
     @ RETURN
rem ***************** Content of data disc ***********
1600 CAT "."&B9$ @ DISP "Purge File" @ INPUT H1$@ PURGE
     H1$ @ GOTO 1000
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP "Correct
     Error condition  +CONT";ERRL
1670 PAUSE
1675 GOTO 500
1690 GOSUB 1700 @ GOTO 1000
rem **************** comment *************************
```

```
1700 DISP "Comment 2 Lines" @ FLIP @ INPUT T1$@ FLIP
1710 T1$=T1$&" " @ IF LEN(T1$)<63 THEN GOTO 1710
1720 RETURN
rem ********* blank subtraction error menu ***********
1800 RESET 7 @ ASSIGN# 1 TO D4$[6,14]&".KSYS"
1820 FOR J=1 TO 20 @ READ# 1 ; Z8@ IF J=3 OR J=4 OR J=6
     OR J=7 OR J=15 OR J=16 THEN GOTO 1840
1835 IF Z8#T(J) THEN BEEP @ GOTO 1885
1840 NEXT J @ Z8=1
1850 FOR K=0 TO T(5)
1860 READ# 1 ; T1(K),T2(K)
1870 T1(K)=-T1(K) @ T2(K)=-T2(K) @ NEXT K @ ASSIGN# 1
     TO *
1875 H1$=D4$[6,14] @ PRINTER IS 2 @ PRINT "Blank:
     ";H1$;" subtracted" @ RETURN
1885 DISP "Variable";J;T(J);"# Blank:";Z8
1890 DISP "Change Variable or (999=ignore)";@ INPUT H1@
     IF H1=999 THEN 1840
1892 IF J=1 OR J=2 OR J=10 OR J=11 OR J=12 OR J=13 OR
     J=19 THEN BEEP @ GOTO 1000 ELSE T(J)=H1
1895 GOTO 1840
rem ************* measure reference ******************
1900 A$="*1" @ GOSUB 7000 @ W(11)=.5 @ A$="*A0.5" @
     GOSUB 7000
1905 A$="*B" @ GOSUB 7000 @ WAIT 2000 @ GOSUB 7000 @
     WAIT 2000
1910 MODE 0,1
1920 REFERENCE 10
1930 IF NMEAS=0 THEN GOTO 1930
1940 MEASURE .1
1950 IF NMEAS=0 THEN GOTO 1950
1960 FOR J=1 TO 10 @ IF VALUE(L(J))>5 THEN GOTO 1900
1970 NEXT J @ C(9)=1 @ RETURN
rem ***************************************************
rem routine for automatic determination of kinetic
     constants
rem **************************************************
rem **************** setup *************************
2000 ERASE STATUS
2020 IF T(14)<T(4)/20 OR T(14)>T(4)/3 THEN BEEP @ DISP
     "[S] wrong !" @ WAIT 3000 @ GOTO 500
2040 GOSUB 3600 @ CLEAR @ PRINTER IS 1 @ H1=0 @ GOSUB
     9510
2050 DISP "1=Exit 2=Input"
2200 GOSUB 3650 @ Z5=0
rem **************** main loop **********************
2230 F7$="0" ! TP
2240 IF Q2=2 THEN GOTO 2247
2245 FOR K=0 TO T(5) @ GOTO 2249
2247 FOR K=T(5) TO 0 STEP -1
2249      Z5=Z5-1 @ IF Z5<0 THEN Z5=0
2250      IF   F7$="2" THEN F7$="0" @ K=P3
2255      FOR J=1 TO 80 @ S3(J)=0 @ NEXT J
2260      R9=0 @ R8=0 @ S6=0 @ S7=0
2300      T(0)=INT(T(20)-K*15/T(5)) @ IF K=0 OR T(0)<0
     OR T(0)>T(20) THEN T(0)=0
2302      T(0)=T(0)+T(8) @ IF T(0)>10 THEN T(0)=10
2305      FOR J=1 TO T(0) @ F6$="0"
2310           GRAPH
2350           GOSUB 3500 ! Execute Assay
2420           GOSUB 4600 ! derivate result
2430           GOSUB 3300 ! linearity check
2450           IF F5$="1" THEN GOTO 2310
2470           FOR J1=2 TO T9/L2+1
2480           S3(J1)=S3(J1)+S1(J1)
2490           NEXT J1 @ R9=R9+S7 @ R8=R8+S6 @ IF
     F6$="1" THEN GOTO 2530
2500      NEXT J
```

```
2530       GOSUB 3700 ! average primary data
2580       IF F6$="1" THEN GOTO 2595
2590       FOR J=2 TO T9/L2+1 @ S3(J)=S3(J)/T(0) @ NEXT
           J @ IF F6$="2" THEN GOSUB 4510
2595       F6=3 @ GOSUB 6500
2600       GOSUB 3800 ! check noise
2620       IF F5$="1" THEN GOSUB 4500
2650       T1(K)=T1(K)+T1(41) @ T2(K)=T2(K)+T2(41) @ IF
                D4$[5,5]="0" THEN 2654
2651       IF ABS(T1(K))>1 OR ABS(T2(K))>1 OR ABS(S9)>1
           THEN PRINT K;T1(K);T2(K);S9 @ GOTO 2654
2652       PRINTER IS 2 @ PRINT USING "DD,X,SD.6D,X,SD.
           6D,X,D.6D" ; K,T1(K),T2(K),S9
2654       IF F7$="1" THEN F7$="0" @ GOTO 3000
           1655 NEXT K
2995 DISP T1$ @ DISP @ DISP @ A$="*WFCC8" @ GOSUB 7000
           @ GOSUB 7700
rem *************** end menu **************************
3000 ON KEY# 3,"Rep.Me" GOTO 3100 @ ON KEY# 2,"Output"
     GOTO 3250 @ ON ERROR GOTO 3000
3010 ON KEY# 8,"Syringe" GOTO 3050 @ ON KEY# 4,"Next
     Pr" GOTO 3030 @ ON KEY# 1,"Exit" GOTO 500
3015 ON KEY# 5,"ResM" GOTO 3060 @ ON KEY# 6,"** End-M"
     GOTO 3000 @ ON KEY# 7,"enu **" GOTO 3000
3017 KEY LABEL @ J=0
rem **** routines performin end menu softkey actions **
3020 J=J+1 @ BEEP 300,10 @ WAIT 2000 @ IF C(6)>1 AND
     J>2 THEN 3250 ELSE 3020
3030 GOSUB 900 @ GOTO 3250
3050 GOSUB 7250 @ GOTO 3000
3060 F7$="2" @ CLEAR @ DISP "Proceed from Point No." @
     INPUT K@ P3=K @ GOSUB 3110
3070 F5$="0" @ GOTO 2240
```

```
3100 F7$="1" @ CLEAR @ DISP "Repeat which Point" @
     INPUT K@  GOSUB 3110 @ GOTO 2255
3110 IF D4$[4,4]#"L" THEN T1(K)=0 @ T2(K)=0 @ GOTO 3200
3120 ASSIGN# 1 TO D4$[6,14]&".KSYS" @ FOR J=1 TO 20 @
     READ# 1 ; T1(41)@ NEXT J
3130 FOR J=0 TO T(5) @ READ# 1 ; Y0,Y1@ IF K=J AND
     F7$="1" THEN T1(K)=-Y0 @ T2(K)=-Y1
3140 IF F7$="2" AND Q2=2 AND K>=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3145 IF F7$="2" AND Q2=1 AND K<=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3150 NEXT J @ ASSIGN# 1 TO *
3200 RETURN
3250 IF F$#"A" OR C(6)<2 THEN 3255
3251 I=B(20) @ IF E(1)=0 THEN H1=W(I+5) @ GOTO 3253
3252 H1=ABS(W(I+5)/E(1)) @ IF I=2 THEN H1=H1*T(3) ELSE
     H1=H1*T(4)
3253 A$="*"&VAL$(I) @ GOSUB 7000 @ W(I+5)=H1 @
     A$="*="&VAL$(H1) @ GOSUB 7000
3254 A$="*E8" @ GOSUB 7000
3255 IF Q1=2 THEN RESET 7 @ H1$=D4$[15,20]&D1$
     &VAL$(D2)&"N" @ GOSUB 1520 @ D4$[4,4]="L"
3270 GOSUB 4200 @ GOTO 1380
rem ***************** linearity check ***************
3300 I3=0 @ S7=0 @ F5$="0" @ ON KEY# 2 GOTO 3390 @ BEEP
     100,10 ! EMTES
3310 FOR J1=T8/L2+1 TO T9/L2+1 @ H1=(S9-S1(J1))^2 @
     I3=I3+1 @ S7=S7+H1
3320 NEXT J1 @ S6=(S7/I3)^.5 @ IF          S6<.00001*
     (1+K/T(5))*B(2) THEN GOTO 3350
3330 IF K=0 THEN GOTO 3350
3340 IF S6>ABS(S9/100*B(2)) THEN DISP
     "Meas";K;J;"Repeated"; Z5 @ BEEP @ F5$="1" @
     Z5=Z5+1
```

```
3345 IF Z5>3*T(8) AND F5$="1" THEN PRINT J;S9;S8;" ??"
     @ F5$="0"
3350 BEEP 200,10 @ OFF KEY# 2 @ RETURN
3390 I6=1 @ GOSUB 4450 @ GOSUB 4550 @ GOTO 3300
rem ************** perform and measure assays ********
3500 T8=INT((B(13)+K/T(5)*(T(6)-B(13)))/L2)*L2
3503 T9=INT((B(14)+K/T(5)*(T(7)-B(14)))/L2)*L2
3505 IF J#1 THEN GOTO 3510 ELSE I=0 @ GOSUB 7900 @ WAIT
     500
3507 IF K<2 OR K=T(5) OR T(5)<7 THEN A$="*H" @ GOSUB
7000 @    WAIT 3000+B(9)*K/T(5)
3510 MODE 0,1 @ A$="*H" @ GOSUB 7000 @ WAIT
     B(8)+B(9)*K/T(5)
3512 H4=(T(11)-T(10))/2+1 @ H2=(T(2)-T(1))/2+1 @
     H3=(T(13)-T(12))/2+1
3514 ERASE MEMORY -1 ! MESSE
3515 ON ERROR GOTO 3597
3520 MEASURE L1,L2,0,T9+L2
3530 FOR J1=1 TO T9/L2+2
3540 IF NMEAS#J1 THEN GOTO 3540
3550 TO MEMORY J1
3560 IF L1>.2 THEN GOSUB 4400
3570 NEXT J1 @ I6=SPOLL(708) @ IF BIT(I6,0)=1 THEN 2995
3575 ALPHA @ IF L2>.2 THEN 3595
3580 FOR J1=1 TO T9/L2+2
3585 RECALL MEMORY J1
3587 GOSUB 4400 @ NEXT J1
3595 RETURN
3597 IF ERRN=2 THEN GOSUB 1900 @ PRINT "Ref. repeated"
     @ GOTO 3500 ELSE GOTO 1660
rem ********** check if data file already exists ******
3600 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     3610
```

```
3605 ASSIGN# 1 TO D$ @ DISP "File exists" @ ASSIGN# 1
     TO * @ BEEP @ WAIT 2000 @ GOTO 1000
3610 IF ERRN=130 OR ERRN=125 THEN DISP B9$;"not found"
     @ BEEP @ WAIT 2000 @ GOTO 3600
3620 ON ERROR GOTO 1660 @ PLOTTER 1 @ PRINTER 2
3625 K=T(5) @ GOSUB 8250 @ L(20)=L(20)*T(14)/H4 @
     C(10)=0
3630 B(6)=T(4)/20 @ GOSUB 4800 @ GOSUB 8000 @ IF
     D4$[5,5]="0" THEN H1=0 ELSE H1=1
3635 GOSUB 9500
3640 FOR K=0 TO 40 @ T1(K)=0 @ T2(K)=0 @ NEXT K
3645 IF D4$[4,4]="L" THEN GOSUB 1800
3647 RETURN
rem ********** print and display headers **************
3650 F6$="0" @ IF D4$[5,5]="1" THEN PRINTER 2 @ PRINT @
     PRINT "NR   Range 1   Range 2  Noise 1"
3655 DISP "Duration";INT(T(8)*T(5)*(T(7)+5) /200+1)*3;
     "Minutes"
3660 DISP "No.      Range 1     Range 2"
3680 RETURN
rem ******* average 2-10 assas ************************
3700 T1(41)=0 @ T2(41)=0 @ H4=W1(3)*T(4)/B(7)
3710 FOR J=1 TO T(0) @ T1(41)=T1(41)+S4(J) @
     T2(41)=T2(41)+S5(J) @ NEXT J
3720 T1(41)=T1(41)/T(0) @ T2(41)=T2(41)/T(0) @
     R9=R9/T(0) @ R9=R9^.5 @ R8=(R8/T(0))^.5
3730 DISP USING "DD,X,5D.DD,XXX,DDD.DD,XXX,.6D" ;
     K,H4.B(3),R8 @ RETURN
rem ******** check for excess noise *******************
3800 S9=0 @ S8=0 @ F5$="0" ! CH
3810 FOR J=1 TO T(0)
3820 S9=S9+(T1(41)-S4(J))^2 @ S8=S8+(T2(41)-S5(J))^2 @
     NEXT J @ S9=(S9/T(0))^.5
3830 IF S9<ABS(.00005*B(1)) OR Q1#1 THEN GOTO 3860
```

```
3850 IF S9>ABS(T1(41)/100*B(1)) THEN DISP "Average";K;"
     repeated";Z5 @ BEEP @ F5$="1"
3855 Z5=Z5+T(0) @ IF Z5>2*T(8) AND F5$="1" THEN 3870
3860 RETURN
3870 PRINT "Average nonreliable" @ FOR J=1 TO T(0) @
     PRINT "Msg";K;J;" ";S4(J);S5(J) @ NEXT J
3875 F5$="0" @ GOTO 3860
rem ********** print and store result *****************
4200 IF D4$[5,5]="0" THEN 4205
4201 PRINTER 2 @ PRINT @ PRINT @ PRINT TAB(7);E4$ @
     GRAPH @ COPY
4202 H1=0 @ GOSUB 9500 @ PRINT T1$ @ ON ERROR GOTO 1660
4205 PRINT @ PRINT @ PRINT "No.  Range 1   Range 2
     [Substr]"
4210 PRINT "    TN(1/sec) TN(1/sec)    uM" @ PRINT
4220 K=T(5) @ GOSUB 8250
4230 A$="PFTN.KSYS"
4250 GOSUB 8500
4260 LORG 0 @ CSIZE 3 @ GOSUB 5500
4270 FOR K=0 TO T(5) @ GOSUB 8250
4290 MOVE H4,H3 @ LABEL "*" @ MOVE H4,J1 @ LABEL "+"
4310 PRINTER IS 2 @ PRINT USING "DD,XXX,SDDD.D,
     XXX,SDDD. D,XX,5D.DD" ; K,H3,J1,H4
4312 PRINT# 1 ; H3,J1,H4
4320 NEXT K @ PRINTER IS 2
4330 PRINT @ IF D4$[5,5]="1" THEN PRINT @ PRINT
     TAB(7);E4$ @ COPY @ PRINT @ PRINT
4350 ASSIGN# 1 TO *
4360 GOSUB 6000 @ PRINT @ PRINT @ PRINT @ RETURN
rem ** get range 1 and 2 corrected for int. reference *
4400 I3=0 @ H1=0
4410 FOR I2=T(1) TO T(2) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4415 S1(J1)=H1/H2 @ H1=0
```

```
4420 FOR I2=T(12) TO T(13) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4425 S2(J1)=H1/H3 @ H1=0
4430 FOR I2=T(10) TO T(11) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4435 S1(J1)=S1(J1)-H1/H4 @ S2(J1)=S2(J1)-H1/H4
4440 RETURN
rem *************** Debug menu ************************
rem
4450 DISP "1=Exit 2=New Stdev 3=Rep.Single" @ DISP
     "4=Rep.Aver. 5=End Me  6=go on"
4455 DISP "7=Test Msg 8=Next Ready";
4460 INPUT H1@ RETURN
4500 IF F$="A" THEN GOTO 2250
4510 GOSUB 4450 @ ON ERROR GOTO 1660 @ I6=2
4520 F6$="0" @ IF H1#3 THEN 4550
4530 F6$="1" @ DISP "No. of Single Measurement" @ INPUT
     J@ FOR J1=1 TO 80 @ S3(J1)=0 @ NEXT J1
4540 R9=R9^2*T(0)/(1.2+T(0)) @ R8=R8^2*T(0)/(1.2+T(0))
     @ GOTO 2310
4550 IF H1=5 THEN 2995
4555 IF H1=7 THEN 2000
4560 IF H1=2 THEN DISP "Max. Nonlin./Noise 2 Inputs";@
     INPUT B(1),B(2)@ GOTO 4590
4565 IF H1=8 THEN GOSUB 900 @ GOTO 4590
4570 IF H1=3 AND I6=1 THEN F6$="2" @ GOTO 4590
4580 IF H1=4 THEN 2250
4590 RETURN
rem ******* derivate absorption vs time **************
4600 S9=0 @ S8=0 @ I1=0
4620 H1=S1(1) @ H2=S2(1)
4630 FOR J1=2 TO T9/L2+1
4635 H3=S1(J1) @ H4=S2(J1)
```

```
4640 S1(J1)=(S1(J1+1)-H1)/(2*L2) @ S2(J1)=(S2(J1+1)-
     H2)/(2*L2) @ IF J1>T9/L2+1 THEN GOTO 4710
4700 IF J1>=T8/L2+1 THEN I1=I1+1 @ S9=S9+S1(J1) @
     S8=S8+S2(J1)
4710 H1=H3 @ H2=H4 @ NEXT J1 @ I3=0 @ S8=S8/I1 @
     S9=S9/I1
4730 DISP USING "DD,X,DD,XX,SZ.6D,XXX,SZ.6D" ;
     K,J,S9,S8
4740 S4(J)=S9 @ S5(J)=S8
4750 RETURN
rem ********** single assay *************************
rem
4800 F6$="0" @ A5=0 @ A6=0 @ MODE 0,1
4810 LAMBDA L(1),L(2),L(3),L(4),L(5),L(6),L(7),
     L(8),L(9),L(10)
4820 TIME SCALE 0 TO T(7)
4830 ABSORBANCE
4832 IF B(3)>T(3)/3 THEN B(3)=T(3)/3
4833 IF B(6)>T(4)/3 THEN B(6)=T(4)/3
4835 W1(3)=B(7)*B(6)/T(4)
4836 W1(2)=B(7)*B(3)/T(3)
4840 IF W1(3)<=0 THEN GOTO 1900
4845 IF C(9)#1 THEN 1000
4850 I=0 @ GOSUB 7980
4860 A$="*H" @ GOSUB 7000
4865 WAIT 5000
4870 J=0 @ K=INT(T(5)/2) @ T9=T(7) @ T8=B(13) @ GOSUB
     3510
4920 H1=0 @ H2=0
4930 FOR J1=1 TO 4 @ H1=S1(J1)+H1 @ H2=S2(J1)+H2 @ NEXT
     J1 @ H1=H1/4 @ H2=H2/4
5050 IF D4$[5,5]#"0" AND A5#1 THEN A$="PFOD.KSYS" @
     GOSUB 8500 @ A5=1
5055 DISP "Initial OD Range 1 and Range 2"
```

```
5056 DISP USING "6X,SD.5D,6X,SD.5D" ; H1,H2 @
     H1=H1/T(15)*1000 @ H2=H2/T(16)*1000
5057 IF ABS(H1)>99999 OR ABS(H2)>99999 THEN DISP
     "[E]";H1;H2;"uM" @ GOTO 5060
5058 DISP USING "3A,X,S5D.3D,4X,S5D.3D,2A" ;
     "[E]",H1,H2,"uM" @ DISP "Deriv. Range1 and Range
     2"
5060 IF D4$[5,5]#"0" THEN F6=1 @ GOSUB 6500 @ F6=2 @
     GOSUB 6500
5150 K=0 @ J=0 @ GOSUB 4600 @ GOSUB 3300 @
     H4=ABS(S6/S9*100) @ IF H4>999 THEN H4=999
5160 DISP USING "14A,X,.6D,3A,3D.D,A" ; "Noise  Range
     1",ABS(S6)," = ",H4,"%" @ H1=0 @ H2=0
5170 FOR J=2 TO 6 @ H1=H1+S1(J) @ NEXT J @ H1=H1/5 @
     A1=H4
5180 FOR J=T(7)/L2-3 TO T(7)/L2+1 @ H2=H2+S1(J) @ NEXT
     J @ H2=H2/5 @ H3=INT(ABS((H1-H2)/H1*100))
5190 DISP "Nonlinearity  : ";H3;"%" @ A2=H3 @ DISP @
     DISP
5300 J1=0 @ IF D4$[5,5]#"0" THEN GOSUB 5400
5310 IF J1=9 THEN 4810
5350 IF A5#2 THEN A$="PFAB.KSYS" @ GOSUB 8500 @ A5=2
5360 F6=1 @ GOSUB 6500 @ F6=2 @ GOSUB 6500
5390 GOSUB 5400 @ IF J1=9 THEN 4810
5395 RETURN
rem ********** softkeys single assay menu ************
5400 OFF KEY# 8 @ ON KEY# 2,"Go on" GOTO 5490 @ ALPHA
5402 ON KEY# 5,"[E]" GOTO 5440 @ ON KEY# 6,"Activ" GOTO
5445      @ OFF KEY# 7
5405 ON KEY# 3,"Graph" GOTO 5420 @ ON KEY# 4,"Alpha"
     GOTO 5430 @ KEY LABEL
5407 IF F$="A" AND C(10)=0 THEN J1=1 ELSE J1=0
5410 IF J1=0 THEN 5410 ELSE BEEP 300,20 @ WAIT 3000 @
     GOTO 5450
```

```
5420 GRAPH @ J1=0 @ GOTO 5410
5430 ALPHA @ J1=0 @ GOTO 5410
5440 DISP B(3);"new";@ INPUT B(3)@ B(5)=B(3) @ J1=9 @
     GOTO 5490
5445 DISP B(10);"new";@ INPUT B(10)@ GOTO 5400
5450 IF B(4)=0 OR Q1=2 THEN 5490
rem ******** automatic adjust of added enzyme *********
5454 IF B(3)<T(3)/200 THEN B(3)=T(3)/200 @ GOTO 5490
5455 IF ABS(S9)>2*B(10) THEN H1=.5 @ GOTO 5492
5460 IF ABS(S9)<B(10)/2 THEN H1=2 @ GOTO 5492
5465 IF ABS(S9)>1.3*B(10) THEN H1=.75 @ GOTO 5492
5470 IF ABS(S9)<.75*B(10) THEN H1=1.25 @ GOTO 5492
5490 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ RETURN
5492 B(3)=B(3)*H1 @ A6=A6+1 @ CLEAR @ DISP
     "Enzymeconc.:"; B(3) @ IF A6>6 THEN J1=0 ELSE J1=9
5493 GOTO 5490
5495 GOSUB 7250 @ GOTO 5400
rem ********* create data file ************************
5500 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     1660
5550 CREATE D$,3+INT(T(5)/10+.5),256
5560 ASSIGN# 1 TO D$
5570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B()
5660 RETURN
rem *********** preform linear regression *************
6000 A1=0 @ A2=0 @ A3=0 @ A4=0 @ A5=0 @ A6=0
6050 FOR K=2 TO T(5) @ GOSUB 8250
6090 IF H3=0 OR H4=0 THEN GOTO 6200 ELSE H1=1/H4 @
     H2=1/H3
6120 A1=A1+1 @ A2=A2+H1 @ A3=A3+H2 @ A4=A4+H1*H2 @
     A5=A5+H1*H1 @ A6=A6+H2*H2
6200 NEXT K
6210 IF A1=0 THEN GOTO 6400
```

```
6220 H1=A5/A1-(A2/A1)^2
6230 H2=A6/A1-(A3/A1)^2
6240 H3=(A4/A1-A2/A1*A3/A1)/H1
6250 H4=A3/A1-H3*A2/A1
6260 H5=H3*H1^.5/H2^.5
6280 H1=ABS(1/(H4/H3)) @ H2=ABS(1/H4)
6290 IF ABS(H1)>9999 OR ABS(H2)>99999 OR ABS(H5)>1 THEN
     GOTO 6400
6300 PRINT USING "15A,4X,6D.2D,X,3A" ; "Velocity
     =",H2,"1/s"
6310 PRINT USING "16A,3X,5D.3D,X,3A" ; "Michaelis
     Const.=",H1," uM"
6320 PRINT USING "14A,9X,3D.4D" ; "Correlation   =",H5
6400 PRINT @ PRINT @ RETURN
rem ********* display graphically ********************
6500 PLOT 0,0,2
6520 FOR J1=2 TO T9/L2+1
6530 H4=(J1-1)*L2
6540 ON F6 GOTO 6550,6560,6570
6550 H3=S1(J1) @ GOTO 6580
6560 H3=S2(J1) @ GOTO 6580
6570 H3=S3(J1)
6580 PLOT H4,H3,1
6590 NEXT J1 @ PENUP @ PLOT 0,0,2 @ RETURN
rem ********* syringe control menu ********************
rem
7000 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF     KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
7005 SET TIMEOUT 7;3000
7010 ON TIMEOUT 7 GOTO 7090
7015 A1$=A$[2,2]
7020 I5=SPOLL(708)
7025 IF BIT(I5,7) THEN WAIT 500 @ GOTO 7020
```

```
7030 IF BIT(I5,5) THEN DISP "Empty" @ F5$="2" @ C(15)=0
     @ GOTO 7082
7032 IF BIT(I5,4) THEN DISP "Full" @ C(15)=0
7035 IF BIT(I5,3) THEN C(15)=0
7040 A$=A$&"," @ OUTPUT 708 :A$
7052 IF A1$="1" THEN M=1
7054 IF A1$="2" THEN M=2
7060 IF A1$="3" THEN M=3
7061 IF A1$="B" OR A1$="D" THEN C(M)=C(M)-W(M+10)
7062 IF A1$="C" THEN C(M)=C(M)+W(M+10)
7063 IF A1$="F" THEN C(M)=0
7064 IF A1$="G" THEN C(M)=W(M)
7065 IF A1$="H" THEN C(1)=C(1)-W1(1) @ C(2)=C(2)-W1(2)
     @ C(3)=C(3)-W1(3)
7070 OFF TIMEOUT 7
7080 RETURN
7082 IF A1$="F" OR A1$="B" OR A1$="D" THEN 7080 ELSE
7040
7090 DISP "Switch on ASSAYOMAT" @ BEEP @ WAIT 5000 @
RESET 7   @ GOTO 7000
rem ************** emulate *K command *****************
7100 DISP A$ @ A$="*TF84A " @ I6=W1(4) @ GOSUB 7200
7110 I6=INT(W1(4)/(W1(1)*48000/W(1)))+257 @ GOSUB 7200
7115 IF W1(2)=0 THEN I6=60258 ELSE I6=INT(W1(4)/
     (W1(2)*48000/W(2)))+257
7120 GOSUB 7200 @ IF W1(3)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(3)*48000/W(3)))+257
7125 GOSUB 7200
7130 GOSUB 7000
7190 RETURN
7200 B$="    " @ H1$="0123456789ABCDEF"
7205 FOR J1=4 TO 1 STEP -1
7210 I=I6 MOD 16 @ B$[J1,J1]=H1$[I+1,I+1] @ I6=I6-I @
     I6=I6/16 @ NEXT J1
```

```
7215 A$=A$&"="&B$[3,4]&"="&B$[1,2]
7220 RETURN
rem *********** softkeys for syringe control menu *****
7250 M=0 @ A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700 @ GOTO
     7480
7280 ON KEY# 1,"Give" GOTO 7430 @ ON KEY# 2,"Suck" GOTO
     7420
7300 ON KEY# 3,"Back" GOTO 7400 @ ON KEY# 4,"Empty"
     GOTO 7460
7320 ON KEY# 5,VAL$(W(M+10)) GOTO 7360
7330 ON KEY# 6,VAL$(C(M)) GOTO 7440
7340 ON KEY# 7,"Retur" GOTO 7355
7350 ON KEY# 8,"Syr."&VAL$(M) GOTO 7480
7352 KEY LABEL @ GOTO 7280
7355 OFF KEY# 1 @ RETURN
7360 CLEAR @ DISP "Volume per Key";@ INPUT H1$@
     W(M+10)=VAL(H1$) @ A$="*A"&H1$ @ GOSUB 7000
7370 GOTO 7280
7400 A$="*D" @ GOSUB 7000 @ GOTO 7280
7420 A$="*C" @ GOSUB 7000 @ GOTO 7280
7430 A$="*B" @ GOSUB 7000 @ GOTO 7280
7440 A$="*G" @ GOSUB 7000 @ GOTO 7280
7460 A$="*F" @ GOSUB 7000 @ GOTO 7280
7480 M=M+1 @ IF M>3 THEN M=1
7485 A$="*"&VAL$(M) @ GOSUB 7000 @ GOTO 7280
rem ******** washing syringe **************************
7500 CLEAR @ DISP "No. of Cycles and Syringe (1-3)";@
     INPUT H1,M
7510 IF M<1 OR M>3 THEN 7500
7520 A$="*"&VAL$(M) @ GOSUB 7000
7530 A$="*E"&VAL$(H1) @ GOSUB 7000
7535 IF H1 MOD 2#0 THEN C(M)=0 ELSE C(M)=W(M+5)
7540 RETURN
rem ********* read information from ASSAYOMATE ********
```

```
7700 SET TIMEOUT 7;2000 @ A$=""
7710 ON TIMEOUT 7 GOTO 7790
7720 ENTER 708 USING "#,B" ; H1@ A$=A$&CHR$(H1) @ IF
     H1=13 OR LEN(A$)>40 THEN 7740
7730 GOTO 7720
7740 OFF TIMEOUT 7 @ ON ERROR GOTO 7790
7750 C(1)=VAL(A$[10,16])
7760 C(2)=VAL(A$[20,26]) @ C(3)=VAL(A$[30,36]) @
C(15)=1
7790 ON ERROR GOTO 1600 @ RETURN
rem ******** define spacing ****************************
7900 IF K=0 THEN N3=0 @ GOTO 7970
7910 ON T(19) GOTO 7920,7930,7950,7940,7960
7920 N3=4*L3^(K-1) @ GOTO 7970
7930 N3=K*300/T(5) @ IF K>T(5)/2 THEN N3=150+INT(K-
     T(5)/2)*600/T(5) @ GOTO 7970 ELSE GOTO 7970
7940 K1=T(5)+1-K @ N3=1/K1*600 @ GOTO 7970
7950 N3=5+K*100/T(5)+3*L3^(K-1) @ GOTO 7970
7960 N3=N3/L(20)
7970 W1(3)=N3*L(20)/10000
7975 W1(2)=B(7)*B(3)/T(3)*(1-B(10)*(T(5)-K)/T(5))
7980 W1(1)=B(7)-W1(2)-W1(3) @ IF I=9 THEN RETURN
7985 A$="*K"&VAL$(W1(1))&","&VAL$(W1(2))&",
     "&VAL$(W1(3)) @ GOSUB 7100
7990 GOSUB 7000 @ RETURN
rem ********* calculate necessary volume **************
8000 S1(1)=4 @ S1(2)=.3 @ S1(3)=.25 @ I=9 @ A$="*WFCC8"
     @ GOSUB
7000 @ GOSUB 7700
8010 FOR K=0 TO T(5) @ GOSUB 7900 @ FOR M=1 TO 3 @
     S1(M)=S1(M)+W1(M)*T(8)*1.1 @ NEXT M @ NEXT K
8060 FOR M=1 TO 3 @ IF S1(M)>C(M) THEN DISP "Syringe
     ";M;" is filled" @ GOSUB 8100
8070 NEXT M @ RETURN
```

```
rem ****************************************************
8100 K=B(20) @ A$="*"&VAL$(M) @ GOSUB 7000 @ IF M#K
     THEN 8120
8105 S1(K)=S1(K)+.2 @ IF S1(K)>W(K) THEN S1(K)=W(K)
8107 IF S1(K)>B(19) THEN S1(K)=B(19)
8110 A$="*="&VAL$(S1(K)) @ GOSUB 7000 @ W(K+5)=S1(K)
8120 A$="*G" @ GOSUB 7000
8190 RETURN
rem ********** calculate enzyme and substrate conc. ***
8200 I=0 ! BER. [E] [S]
8205 GOSUB 7900
8210 H2=T(3)*W1(2)/B(7)
8220 H4=T(4)*W1(3)/B(7) ! [S]
8230 H3=T1(K)*1000000/T(17)/H2 @ J1=T2(K)*1000000
     /T(18)/H2
8240 RETURN
8250 I=9 @ GOSUB 8205
8260 RETURN
rem ********read axes from file and display **********
8500 RESET 7
8510 ASSIGN# 1 TO A$ @ READ# 1 ; H3,H4,X0,X1,X2,X3,Y0,
     Y1,Y2,Y3,E3$,E4$
8550 GRAPH @ GCLEAR @ SCALE 0,100,0,100 @ IF X2=0 THEN
     X2=(X1-X0)/5
8560 MOVE 30,0 @ LDIR 0 @ CSIZE 3 @ LABEL E3$ @ F5=(X1-
     X0)/7 @ F6=(Y1-Y0)/10
8590 SCALE X0-F5,X1,Y0-F6,Y1 @ XAXIS Y1,X3,X0,X0+X3 @  X
     AXIS Y1,X2,X0+X3,X1
8620 XAXIS Y0,X3,X0,X0+X3 @ XAXIS Y0,X2,X0+X3,X1 @
     YAXIS X1,Y3,Y0,Y0+Y3
8650 YAXIS X1,Y2,Y0+Y3,Y1 @ YAXIS X0,Y3,Y0,Y0+Y3 @
     YAXIS X0,Y2,Y0+Y3,Y1
8660 FOR K=X0+X3 TO X1 STEP X2
8670 MOVE K-F5/5,Y0-(Y1-Y0)/20
```

```
8680 LABEL VAL$(K) @ NEXT K
8690 FOR K=Y0+Y3 TO Y1 STEP Y2
8700 MOVE X0-(X1-X0)/10,K-F6/2
8710 LABEL VAL$(K) @ NEXT K
8900 ASSIGN# 1 TO * @ RETURN
rem ******* print header ******************************
9500 PRINT @ PRINT @ PRINT
9510 PRINT "Measurement ";D4$[15,20];"  ";D1$;D2 @
     PRINT "Diskette:";B9$;"  ";D3$
9540 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9542 IF H1=0 THEN 9700
9545 PRINT "Time    [S]min";TAB(16);B(13);"bis";
     TAB(24);B(14);"sec"
9546 PRINT "Time    [S]max";TAB(16);T(6);"bis";
     TAB(24);T(7);"sec"
9547 PRINT "Interval  ";L1;" / Integration";L2
9548 PRINT "Max.Stdev";B(1);" / Nonlin.";B(2);"%"
9550 PRINT "Range 1 from   ";T(1);" to";T(2);"nm"
9555 PRINT "Range 2 from   ";T(12);" to";T(13);"nm"
9557 PRINT "Int. Ref. from ";T(10);" to";T(11);"nm"
9560 PRINT "[";E1$;"]";TAB(23);T(3);"nM"
9565 PRINT "in the Assay :  ";B(10);TAB(23);B(3);"nM"
9580 PRINT "[";E2$;"]";TAB(23);T(4);"uM"
9585 PRINT "Max. Conc. in Assay:  ";T(14);"uM"
9600 PRINT "No. of Points";T(5);TAB(24);"Rep.";T(8)
9650 PRINT "Delta Epsilon 1 :  ";T(17);"1/mM cm" @
     PRINT "Delta Epsilon 2 :  ";T(18);"1/mM cm"
9657 PRINT "Assay Volume :";TAB(25);B(7);"ml"
9690 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9700 PRINTER IS 2 @ RETURN

rem ***************************************************
rem *******   MICHKIN software :  dacom.ksys    *******
rem ********   Author : Bruno Michel              *****
rem ***************************************************
rem
rem
rem
rem **************** 50 ! DACOM   B.MICHEL 05.11.86
rem **************** 60 !            KSYS
rem
rem The functions of this part of the MICHKIN software
     are :
rem
rem a) data transfer to host computer
rem b) display and edit of data files and header
     information
rem c) averageing blanks
rem d) copying files
rem e) plotting data to HP Series 74 Plotter
rem
rem ***************************************************
rem                 start of source code
rem ***************************************************
rem **************** common data area ****************
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
     D3$[20],D4$[20],B9$[8]
rem ************** global variables and buffers ******
150 DIM Z$[2000],B8$[64],H1$[64]
160 IOBUFFER Z$
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],
     E5$[60],N9$[30]
190 SHORT Y0,Y1,Y2,Y3,X0,X1,X2,X3,L1,L2,S4(41),Q1,Q2
195 FOR K=0 TO 41 @ S4(K)=0 @ NEXT K @ I9$=" "
```

```
200 F6$="0" @ Q1=1 @ Q2=1 @ F7$="0" @ G1=1 @ G2=1 @
     W9=100 @ CLEAR
205 E3$="..." @ E4$="..." @ X0=0 @ X1=1 @ X2=1 @ X3=0 @
     Y0=0 @ Y1=1 @ Y2=1 @ Y3=0 @ B8$=" "
207 FOR I=1 TO 63 @ B8$=B8$&" " @ NEXT I @
     E5$="...........
210 ERASE STATUS
220 ON ERROR GOTO 250
230 FOR J=1 TO 99
240 ERASE STANDARD J
250 NEXT J @ ON ERROR GOTO 320
255 PLOTTER 1
rem ******test if common data have been initialized ***
310 IF C(8)=1 THEN GOTO 1000
320 CHAIN "Autost.KSYS"
rem ************** check variables menu ***************
rem ************** average blanks and copy files ******
500 L1=B(11) @ L2=B(12) @ ON ERROR GOTO 500 @ CLEAR @
     DISP "**** Display Variables Menu ****"
510 CLEAR @ DISP "Duration ";L(11);L(12);"and";
     T(6);T(7);"sec"
520 DISP "Range  1 from ";T(1);" to  ";T(2) @ DISP
     "Range  2 from ";T(12);" to  ";T(13)
540 DISP "Int. Ref. from ";T(10);" to  ";T(11)
550 DISP E1$;TAB(20);T(3);"nm (E)" @ DISP
     E2$;TAB(20);T(4);"um (S)"
570 DISP "[S] Assay";T(14);"/ Stdev";B(1);"%"
580 DISP "No.of Rep.";T(8);" / No.of Points";T(5)
590 DISP "dEps1";T(17);"dEps2";T(18);"1/mM cm"
610 IF T(19)=1 THEN DISP "Geom. Row"
620 IF T(19)=2 THEN DISP "Arithm. Row"
630 IF T(19)=3 THEN DISP "Mixed Row"
635 IF T(19)=4 THEN DISP "Reciprocal Row"
640 DISP T1$
```

```
rem *************** softkeys ***************************
650 ON KEY# 1,"Exit" GOTO 1000
655 ON KEY# 2,"ReadV" GOTO 690
660 ON KEY# 3,"Dacom" GOTO 2500
665 ON KEY# 4,"Comme" GOTO 780
670 ON KEY# 5,"StorB" GOTO 1500
675 ON KEY# 6,"ReadB" GOTO 1800
680 ON KEY# 7,"CopyF" GOTO 900
685 ON KEY# 8,"Avg Bl" GOTO 3500
686 KEY LABEL @ GOTO 650
690 GOSUB 700 @ GOTO 500
rem ******* routines performing softkey functions ****
700 CLEAR @ DISP "Name of Variable file";@ INPUT D$@
     D$=D$&".KSYS" @ ASSIGN# 1 TO D$
720 READ# 1 ; D1$,D2,D4$,D4$,E1$,E2$,T1$,T(),L(),B()
730 IF D4$[1,1]#"K" THEN DISP "No Kinetic Variable
     file" @ BEEP @ GOTO 700
745 RETURN
750 CLEAR @ DISP "Are you sure to initialize ? ";@
     INPUT H1$[1,1]
755 IF H1$[1,1]="J" THEN INITIALIZE B9$,":D701",20 @
     GOTO 1000 ELSE GOTO 1000
760 IF Q1=1 THEN Q1=2 ELSE Q1=1
765 GOTO 500
780 CLEAR @ FLIP @ DISP "New Comment" @ INPUT T1$@ FLIP
     @ GOTO 500
800 DISP "The Mainprogram is loaded       Please wait
     about 20 sec" @ DISP @ BEEP
810 CHAIN "Autost.KSYS"
900 ! COPY FILE
910 CLEAR @ DISP "Enter name of Sourcefile and
     Drivetype ':d7xx'" @ INPUT D$
920 DISP "Enter name of Destinationfile   and Drivetype
     ':d7xx'" @ INPUT Z$
```

```
930 COPY D$ TO Z$
940 GOTO 500
rem ************ datatransfer main menu ***************
rem
1000 CLEAR @ DISP @ DISP "*********** DACOM
     *************" @ ON ERROR GOTO 1000
1010 DISP "           MAIN MENU" @ DISP "**************
     *****************" @ DISP @ DISP
1050 DISP "Please select a softkey:"
1100 ON KEY# 1,"D Var" GOTO 500
1110 ON KEY# 2,"Dacom" GOTO 2500
1120 ON KEY# 3,"Direct" GOTO 1600
1130 ON KEY# 4,"PlotM" GOTO 7000
1140 ON KEY# 5,"     " GOTO 1000
1150 ON KEY# 6,"DCorr" GOTO 2000
1160 ON KEY# 7,"MAINPG" GOTO 800
1170 ON KEY# 8,"      " GOTO 1000
1190 KEY LABEL @ GOTO 1100
rem ******** routines performing softkey functions ***
1300 GOSUB 5000 @ GOTO 1000
1310 GOSUB 5500 @ GOTO 1000
1500 CLEAR @ DISP "Name of Blankfile (9 Characters)" @
     INPUT H1$@ IF LEN(H1$)#9 THEN GOTO 1500
1510 H2$=H1$&".KSYS"
1540 CREATE H2$,5,256 @ ASSIGN# 1 TO H2$
1560 FOR J=1 TO 20 @ PRINT# 1 ; T(J) @ NEXT J
1570 FOR J=0 TO T(5) @ PRINT# 1 ; T1(J),T2(J) @ NEXT J
1580 ASSIGN# 1 TO *
1590 GOTO 1000
1600 CLEAR @ DISP "Please select Drive 1=700  2=701
     3=710 ";@ INPUT H1
1605 IF H1=1 THEN MASS STORAGE IS ":D700" @ GOTO 1610
1606 IF H1=2 THEN MASS STORAGE IS ":D701" ELSE MASS
     STORAGE IS ":D710"
```

```
1610 CLEAR
1615 CAT @ DISP "Enter name of File to be Purged add.
     Functions: PACK)" @ INPUT H1$
1620 IF H1$="PACK" THEN PACK @ GOTO 1000
1625 IF H1$="N" OR H1$="" THEN GOTO 1000
1630 PURGE H1$ @ GOTO 1000
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP "Correct
     errorcondition    +CONT";ERRL
1670 PAUSE
1700 DISP "Comment :";@ FLIP @ INPUT T1$@ PRINT T1$ @
     FLIP @ GOTO 1000
1800 CLEAR @ DISP "Name of Blankfile (9 Characters)" @
     INPUT H1$@ IF LEN(H1$)#9 THEN GOTO 1800
1805 H2$=H1$&".KSYS"
1810 ASSIGN# 1 TO H2$
1820 FOR J=1 TO 20 @ READ# 1 ; T1(41)
1840 NEXT J
1850 FOR K=0 TO T(5)
1860 READ# 1 ; T1(K),T2(K)
1875 NEXT K @ GOTO 1000
rem *********** data edit menu ************************
rem
2000 R1=0
2005 CLEAR @ DISP "******* Data edit Menu ********" @
     DISP "No. Range 1  Range 2   [Substr]"
2010 FOR K=R1*10 TO 9+R1*10
2015 DISP USING "DD,X,SDDD.DDDDD,X,SDDD.DDDDD,
     X,SDDD.DD" ; K,T1(K),T2(K),S4(K)
2050 NEXT K
2100 ON KEY# 1,"Exit" GOTO 1000
2110 ON KEY# 2,"NexP" GOTO 2300
2120 ON KEY# 3,"FormP" GOTO 2310
2130 ON KEY# 4,"INPUT" GOTO 2400
2140 ON KEY# 5,"INKo" GOTO 2450
```

```
2150 ON KEY# 6,"ReadD" GOTO 2200
2160 ON KEY# 7,"StoD" GOTO 2210
2170 ON KEY# 8,"PlotM" GOTO 7000
2190 KEY LABEL @ GOTO 2100
rem ********* routines performing edit functions ******
2200 GOSUB 5000 @ GOTO 2000
2210 GOSUB 5500 @ GOTO 2000
2300 R1=R1+1 @ IF R1>4 THEN R1=4
2305 GOTO 2005
2310 R1=R1-1 @ IF R1<0 THEN R1=0
2315 GOTO 2005
2400 CLEAR @ DISP "Number ,Column (1-3)";@ INPUT K,H1
2410 DISP "Value";
2412 IF H1=1 THEN INPUT T1(K)
2415 IF H1=2 THEN INPUT T2(K)
2420 IF H1=3 THEN INPUT S4(K)
2430 GOTO 2005
2450 CLEAR @ DISP "Column (1-3)";@ INPUT H1
2460 FOR K=0 TO T(5)
2470 DISP "Value";K;
2472 IF H1=1 THEN INPUT T1(K)
2475 IF H1=2 THEN INPUT T2(K)
2480 IF H1=3 THEN INPUT S4(K)
2490 NEXT K @ GOTO 2005
2495 GOSUB 5000 @ GOSUB 4400 @ GOTO 500
rem ********** data communication subroutine **********
rem
2500 IF LEN(I9$)#2 THEN CLEAR @ DISP "Please enter
     initials" @ INPUT I9$@ GOTO 2500
2510 ! ***** DACOM ************
2520 ! ************************
2530 CLEAR @ DISP @ DISP "*********************
     **********"
2535 DISP "*****  DATA TRANSFER  *********" @
```

```
          DISP "********************************"
2540 DISP @ DISP "wait ";W9;" msec"
2550 GOSUB 3000 ! SETUP CONTR
2555 OFF KEY# 5 @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
2560 ON KEY# 1,"Exit" GOTO 1000
2565 ON KEY# 2,"Abort" GOTO 2500
2570 ON KEY# 3,"SEND " GOTO 2585
2575 ON KEY# 4,"wait " GOTO 3300
2577 ON KEY# 5,"D Var" GOTO 500
2580 KEY LABEL @ GOTO 2555
2585 ON ERROR GOTO 2780
2590 CLEAR @ CAT "."&B9$
2600 DISP "Enter Filename of File to be    transferred
     (Memory=already in Memory"
2605 FLIP @ INPUT H1$@ IF LEN(H1$)>10 THEN D$=H1$[1,10]
     ELSE D$=H1$
2610 D$=D$&"."&B9$ @ FLIP @ IF D$[1,6]#"Memory" THEN
     GOSUB 5050
2620 SET TIMEOUT 10;20000
2630 ON TIMEOUT 10 GOTO 2780
2640 PRINTER IS 10
2650 DISP "Please enter filename on host (8
     characters)";
2652 FLIP @ INPUT B8$
2655 IF LEN(B8$)>10 AND B8$[2,2]=":" THEN H1$=B8$[1,10]
     @ GOTO 2660
2657 IF LEN(B8$)>8 THEN H1$=B8$[1,8] ELSE H1$=B8$
2660 FLIP @ H1$="OPEN "&H1$&".K"&I9$
2665 IF LEN(H1$)<30 THEN H1$=H1$&" " @ GOTO 2665
2670 Z$="" @ OUTPUT Z$ USING "30A" ; H1$ @ DISP "OPEN
     FILE"
2675 TRANSFER Z$ TO 10 INTR
2740 DISP "FILE ";D$;" IS TRANSFERRED  " @ ENTER 10 ;
     H1$@ DISP H1$ @ WAIT 25*W9
```

```
2750 GOSUB 4400 ! TRANSFER
2760 PRINT "CLOSE" @ PRINTER IS 2 @ DISP "FILE CLOSED "
     @ DISP "READY"
2770 GOTO 2555
2780 BEEP @ DISP "ERROR DURING DATA TRANSFER" @ OFF
     TIMEOUT 10 @ WAIT 3000 @ RESET 10
2790 GOTO 2500
2800 TRANSFER Z$ TO 10 INTR
2820 RETURN
rem ********** recover from error ********************
2900 IF ERRN=67 AND ERRL=5060 THEN PRINT "File ";D$;"
     not found" @ GOTO 2930
2905 IF ERRN=130 THEN PRINT "Diskette";B9$;" not found"
     @ GOTO 2930
2910 IF ERRN=129 THEN PRINT "Diskette damaged" @ GOTO
     2930
2925 PRINT "ERROR NUMBER ";ERRN;" ON LINE   ";ERRL @
     PRINT "NEXT TRY     "
2930 CLEAR @ DISP "ERROR DURING DATATRANSFER" @ WAIT
2000 @    GOTO 2600
2990 ! ************************
3000 ! SETUP CONTROL
3010 CONTROL 10,2 ; 7 ! SET MODEM LINES
3020 CONTROL 10,3 ; 15 ! 9600 BAUD
3030 CONTROL 10,4 ; 7 ! 8 BITS 2 STOP BITS NO PARITY
3040 CONTROL 10,5 ; 48 ! HANDS
3050 CONTROL 10,11 ; 192
3060 CONTROL 10,14 ; 19
3070 CONTROL 10,15 ; 17
3090 SET TIMEOUT 10;10000
3100 ON TIMEOUT 10 GOTO 3200
3110 PRINTER IS 10
3120 PRINT "HELLO"
3130 H1$="" @ ENTER 10 ; H1$
```

```
3140 IF H1$#"hello" THEN GOTO 1000
3150 OFF TIMEOUT 10 @ PRINTER IS 2
3190 RETURN
rem ************* solve data transfer problems ********
3200 DISP "Host doesn't respond" @ CRT IS 1
3210 DISP "Please check  :                    a) if
     datatransfer program on      host is running"
3220 DISP "b) if RS-232 cable is plugged in
correctly"
3230 DISP "c) if handshake protocol is        correct
     (see manual chapt. 4 )"
3240 DISP "Press (cont) key" @ PAUSE
3290 OFF TIMEOUT 10 @ RESET 10 @ GOTO 1000
3300 CLEAR @ DISP "Please enter waiting time in ms
     after sending of one line" @ INPUT W9
3330 GOTO 2500
rem *****************************************************
rem *************** routine for averaging blanks *****
3500 REM ***MITTEL NULLINIE ***
3510 FOR I=0 TO 41 @ T1(I)=0 @ T2(I)=0 @ NEXT I
3520 CLEAR @ DISP "Enter number of Blanks to be
     averaged  (1-5)" @ INPUT A1
3530 IF A1>5 OR A1<0 THEN GOTO 3520
3540 CAT ".KSYS"
3550 FOR I=1 TO A1
3555 DISP "Enter name of Blank";I;"(9 Characters)"
3560 INPUT H1$@ IF LEN(H1$)#9 THEN GOTO 3555
3570 N9$[I*9-8,I*9]=H1$
3580 NEXT I @ ON ERROR GOTO 3550
3590 FOR I=1 TO A1
3600 ASSIGN# 1 TO N9$[I*9-8,I*9]&".KSYS"
3610 FOR J=1 TO 20 @ READ# 1 ; Z8
3620 IF I=1 THEN T(J)=Z8
```

```
3630 IF Z8#T(J) THEN DISP "Variable";J;"inkompatible" @
     PAUSE
3640 NEXT J
3645 ON ERROR GOTO 1660
3650 FOR J=0 TO T(5)
3660 READ# 1 ; Z8@ T1(J)=T1(J)+Z8
3670 READ# 1 ; Z8@ T2(J)=T2(J)+Z8
3680 NEXT J @ ASSIGN# 1 TO *
3690 NEXT I
3692 FOR J=0 TO T(5)
3694 T1(J)=T1(J)/A1
3696 T2(J)=T2(J)/A1
3698 NEXT J
3700 Z8=1 @ GOTO 1500
rem *********** plottting subroutine *****************
4200 H2$="0" ! AUSGABE
4210 GOSUB 8500 @ IF H2$="1" THEN GOSUB 6000
4260 LORG 0 @ CSIZE 3
4270 FOR K=1 TO T(5) @ IF G2=1 THEN GOTO 4271 ELSE GOTO
     4272
4271 IF T1(K)=0 OR S4(K)=0 THEN GOTO 4370 ELSE GOTO
     4280
4272 IF T2(K)=0 OR S4(K)=0 THEN GOTO 4370
4280 ON G1 GOTO 4310,4320,4330,4340
4310 X=S4(K) @ IF G2=1 THEN Y=T1(K) ELSE Y=T2(K)
4315 GOTO 4350
4320 X=1/S4(K) @ IF G2=1 THEN Y=1/T1(K) ELSE Y=1/T2(K)
4325 GOTO 4350
4330 IF G2=1 THEN X=T1(K) @ Y=T1(K)/S4(K) ELSE X=T2(K)
     @ Y=T2(K)/S4(K)
4335 GOTO 4350
4340 X=S4(K) @ IF G2=1 THEN Y=S4(K)/T1(K) ELSE
     Y=S4(K)/T2(K)
4350 MOVE X,Y @ IF G2=1 THEN LABEL "*" ELSE LABEL "+"
```

```
4360 IF H2$="1" THEN GOSUB 6700
4370 NEXT K @ MOVE X0+X2,Y1-Y2 @ LABEL "Key1= Exit
     Key2=Papier Plot"
4380 ON KEY# 1 GOTO 1000 @ ON KEY# 2 GOTO 4390 @ GOTO
4380
4390 H2$="1" @ GOTO 4210
rem ******* transfer header ***************************
rem ************ transfer data   *********************
4400 ! PUT VAL IN Z$ *********
4405 Z$=""
4410 OUTPUT Z$ USING "#,A,DDD,18A,20A,20A,20A,64A" ;
     D1$,D2,D3$,D4$,E1$,E2$,T1$
4420 GOSUB 4800 @ WAIT 9*W9
4440 FOR K=1 TO 15 @ OUTPUT Z$ USING "#,S5D.5D" ; T(K)
     @ NEXT K
4445 FOR K=1 TO 5 @ OUTPUT Z$ USING "#,S5D.5D" ; B(K) @
     NEXT K @ GOSUB 4800 @ WAIT 9*W9
4450 OUTPUT Z$ USING "#,3A" ; "   " @ GOSUB 4800
4460 OUTPUT Z$ USING "#,3A" ; "   " @ GOSUB 4800
4480 FOR K=0 TO T(5)
4485 OUTPUT Z$ USING "#,SZ.4DE,SZ.4DE,SZ.4DE" ;
     T1(K),T2(K),S4(K)
4500 GOSUB 4800
4510 NEXT K
4520 IF T(9)#3 THEN RETURN
4525 L1=B(11)
4530 I=0 @ FOR K=1 TO T(5)
4540 FOR J=1 TO T(7)*2/L1+2
4550 READ# 1 ; S1(1)@ OUTPUT Z$ USING "#,SZ.5D" ; S1(1)
4560 I=I+1 @ IF I=16 THEN GOSUB
4800 @ I=0 @ WAIT W9
4570 NEXT J @ NEXT K @ OUTPUT Z$ USING "#,S7D" ; 99999
     @ GOSUB 4800
4600 ASSIGN# 1 TO * @ RETURN
```

```
4800 ! AUSGABEROUTINE
4810 TRANSFER Z$ TO 10 INTR
4820 Z$="" @ WAIT W9
4830 RETURN
rem ************ read data from HP85 file *************
5000 ! READ DATA
5010 CLEAR @ CAT "."&B9$ @ DISP "Filename";@ INPUT D$@
     D$=D$&"."&B9$ @ GOSUB 5050 @ RETURN
5050 ! ******LESE VAR UND DATEN
5060 ASSIGN# 1 TO D$
5070 READ# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B()
5080 FOR K=0 TO T(5)
5100 READ# 1 ; H1,H2,H3@ T1(K)=H1 @ T2(K)=H2 @ S4(K)=H3
5120 NEXT K
5150 IF T(9)#3 THEN ASSIGN# 1 TO *
5160 RETURN
5500 ! STORE DATA
5510 CLEAR @ CAT "."&B9$ @ DISP "Filename";@ INPUT D$
5550 CREATE D$,10,256
5560 ASSIGN# 1 TO D$
5570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B()
5580 FOR K=0 TO T(5)
5600 H1=T1(K) @ H2=T2(K) @ H3=S4(K) @ PRINT# 1 ;
     H1,H2,H3
5620 NEXT K
5650 ASSIGN# 1 TO *
5660 RETURN
rem ************* plot axes ****************************
6000 ! AXES
6020 P3=2*400 @ P4=1*400 @ P5=(2+25)*400 @
     P6=(1+16)*400
6040 PRINTER IS 705 @ H1$=CHR$(3) @              D$=D4
     $[17,20]&D1$&VAL $(D2)&" " @ P1=.000001 @
     P2=.000001
```

```
6050 F5=X1-X0 @ F6=Y1-Y0
6060 IF ABS(F5*P1)<1000 THEN P1=P1*10 @ GOTO 6060
6065 IF ABS(F6*P2)<1000 THEN P2=P2*10 @ GOTO 6065
6067 F5=(P5-P3)/100 @ F6=(P6-P4)/100
6070 PRINT "IN;SP1;"
6080 PRINT "IP";P3;",";P4;",";P5;",";P6;";"
6085 PRINT "SC0,10000,0,10000;"
6087 PRINT "PA4000,9500;"
6089 PRINT "DT";H1$;";"
6090 PRINT "SI.3,.4;CP0,0;LB";E5$;H1$;";"
6100 PRINT "PA4000,100;SI.25,.3;LB";E3$;H1$;";"
6110 PRINT "PA100,3000;DI0,1;LB";E4$,H1$;";DI1,0;"
6120 PRINT "PU;PA 1300,1300;PD;PA
1300,9300;PA9300,9300,9300,1300,1300,1300;PU;"
6150 FOR I1=X0+X3 TO X1 STEP X2
6155 I2=INT((I1-X0)*8000/(X1-X0)+1300)
6160 PRINT "PU;PA";I2;",1300;PD;XT;PU;"
6165 PRINT "SI.2,.25;CP-1,-1.2;LB";VAL$(I1);H1$;";" @
     NEXT I1
6170 FOR I1=Y0+Y3 TO Y1 STEP Y2
6175 I2=INT((I1-Y0)*8000/(Y1-Y0)+1300)
6180 PRINT "PA9300,";I2;";PD;YT;PU;" @ NEXT I1
6200 FOR I1=Y0+Y3 TO Y1 STEP Y2
6205 I2=INT((I1-Y0)*8000/(Y1-Y0)+1300)
6210 PRINT "PU;PA1300,";I2;";PD;YT;PU;"
6215 PRINT "CP-5,-.5;LB";I1;H1$;";" @ NEXT I1
6220 FOR I1=X0+X3 TO X1 STEP X2
6225 I2=INT((I1-X0)*8000/(X1-X0)+1300)
6230 PRINT "PU;PA";I2;",9300;PD;XT;PU;" @ NEXT I1
6240 P3=INT(P3+F5*13)-50 @ P4=INT(P4+F6*13) @
     P5=INT(P5-F5*7)-470 @ P6=INT(P6-F6*7)
6250 PRINT "IP";P3;",";P4;",";P5;",";P6;";"
6255 PRINT "IW":P3;",";P4;",":P5;",";P6;";"
```

```
6260 H1=INT(X0*P1) @ H2=INT(X1*P1) @ H3=INT(Y0*P2) @
     H4=INT(Y1*P2)
6300 PRINT "SC";VAL$(H1);",";VAL$(H2);",";
     VAL$(H3);",";VAL$(H4);";" @ RETURN
6500 PRINTER IS 705 @ H2=J @ PRINT "PU;" @ GOSUB 6900
6520 FOR J1=2 TO T(7)/L2+1
6530 H4=INT((J1-1)*L2*P1)
6540 ON H1 GOTO 6550,6560,6570
6550 H3=INT(S1(J1)*P2) @ GOTO 6580
6560 H3=INT(S2(J1)*P2) @ GOTO 6580
6570 H3=INT(S3(J1)*P2)
6580 PRINT "PA";H4;",";H3;";PD;" @ NEXT J1 @ PRINT
     "PU;"
6590 RETURN
6700 PRINTER IS 705 @ PRINT "SP 1;"
6705 H1=INT(Y*P2) @ H4=INT(X*P1)
6710 IF G2=1 THEN H6$="*"&CHR$(3) @ PRINT
     "PU;PA";H4;",";H1;";LB";H6$;";"
6720 IF G1=2 THEN H6$="+"&CHR$(3) @ PRINT
     "PU;PA";H4;",";H1;";LB";H6$;";"
6730 IF K=T(5) THEN PRINT "PU;PA0,0;"
6740 RETURN
6900 IF H2=1 THEN PRINT "LT;"
6910 IF H2=2 THEN PRINT "LT1,1;"
6920 IF H2=3 THEN PRINT "LT2,1;"
6930 IF H2=4 THEN PRINT "LT3,2;"
6940 IF H2=5 THEN PRINT "LT4,4;"
6950 PRINT "SP";H3;";" @ RETURN
rem ***********MICHKIN plot menu **********************
rem
7000 CLEAR @ DISP "*********** DACOM *************" @
     ON ERROR GOTO 7000
7010 DISP "             PLOT MENU" @ DISP
     "********************************" @ DISP
```

```
7020 DISP "Title of Plot :" @ DISP E5$
7030 DISP "X-Axis ";X0;X1;X2;X3 @ DISP "X-String ";E3$
7040 DISP "Y-Axis ";Y0;Y1;Y2;Y3 @ DISP "Y-String ";E4$
7050 IF G1=1 THEN DISP "Tn vs [Substr] Plot"
7055 IF G1=2 THEN DISP "Lineweaver Burke Plot"
7060 IF G1=3 THEN DISP "Eadie Hofstee Plot"
7070 IF G1=4 THEN DISP "Hannes Wolfe Plot"
7080 IF G2=1 THEN DISP "Range  1 " ELSE DISP " Range 2"
7100 ON KEY# 1,"Exit" GOTO 1000
7110 ON KEY# 2,"ReaD" GOTO 7300
7120 ON KEY# 3,"Inhalt" GOTO 1600
7130 ON KEY# 4,"Page2" GOTO 7200
7140 ON KEY# 5,"Title" GOTO 7360
7150 ON KEY# 6,"DKorr" GOTO 2000
7160 ON KEY# 7,"     " GOTO 800
7170 ON KEY# 8,"     " GOTO 7100
7190 KEY LABEL @ GOTO 7100
7200 ON KEY# 1,"BerNR" GOTO 7350
7210 ON KEY# 2,"StoD" GOTO 7310
7220 ON KEY# 3,"Plot D" GOTO 4200
7230 ON KEY# 4,"Page1" GOTO 7100
7240 ON KEY# 5,"    " GOTO 7000
7250 ON KEY# 6,"X-AX" GOTO 7320
7260 ON KEY# 7,"Y-AX" GOTO 7330
7270 ON KEY# 8,"pl A" GOTO 7340
7290 KEY LABEL @ GOTO 7200
rem ******* routines performing plot menu commands ****
7300 GOSUB 5000 @ GOTO 7000
7310 GOSUB 5500 @ GOTO 7000
7320 CLEAR @ DISP "Enter parameters for X-Axis  (4e)";@
     INPUT X0,X1,X2,X3
7325 FLIP @ DISP "Enter X-String  ";@ INPUT E3$@ FLIP @
     GOTO 7000
```

```
7330 CLEAR @ DISP "Enter Parameters for Y-Axis  (4e)";@
     INPUT Y0,Y1,Y2,Y3
7335 FLIP @ DISP "Enter Y-String ";@ INPUT E4$@ FLIP @
     GOTO 7000
7340 G1=G1+1 @ IF G1>4 THEN G1=1
7345 GOTO 7000
7350 IF G2=1 THEN G2=2 ELSE G2=1
7355 GOTO 7000
7360 CLEAR @ FLIP @ DISP "Enter Title of Plot  ";@ I
     NPUT E5$@ FLIP
7365 GOTO 7000
rem ********** display axes and data routine ********
7370 ! *************************
8500 GRAPH @ GCLEAR @ DEG
8550 SCALE 0,100,0,100 @ IF X2=0 THEN X2=(X1-X0)/5
8560 MOVE 30,0 @ LDIR 0 @ CSIZE 3 @ LABEL E3$ @ F5=(X1-
     X0)/7 @ F6=(Y1-Y0)/10
8590 SCALE X0-F5,X1,Y0-F6,Y1
8600 XAXIS Y1,X3,X0,X0+X3
8610 XAXIS Y1,X2/2,X0+X3,X1
8620 XAXIS Y0,X3,X0,X0+X3
8630 XAXIS Y0,X2/2,X0+X3,X1
8640 YAXIS X1,Y3,Y0,Y0+Y3
8650 YAXIS X1,Y2/2,Y0+Y3,Y1
8660 YAXIS X0,Y3,Y0,Y0+Y3
8670 YAXIS X0,Y2/2,Y0+Y3,Y1
8730 FOR K=X0+X3 TO X1  STEP X2
8740 MOVE K-F5/5,Y0-(Y1-Y0)/20
8750 LABEL VAL$(K) @ NEXT K
8780 FOR K=Y0+Y3 TO Y1  STEP Y2
8790 MOVE X0-(X1-X0)/10,K-F6/2
8800 LABEL VAL$(K) @ NEXT K
8830 RETURN
rem *************** print header ********************
```

```
9500 PRINTER IS 2 @ PRINT @ PRINT @ PRINT @ PRINT
     "Messung  ";D1$;"  ";D2;"     ";D4$[17,20]
9530 PRINT "Datum  ";D3$
9540 PRINT "--------------------------------" @ PRINT
9545 PRINT "Messdauer von";TAB(15);T(6);"
     bis";TAB(24);T(7);"sec"
9546 PRINT "Messdauer2von";TAB(15);L(11);"bis";
     TAB(24);L(12);"sec"
9547 PRINT "Intervall ";L1;" / Integration";L2
9548 PRINT "Max. relative Standardabw ";B(1);"%"
9550 PRINT "Bereich 1 von    ";T(1);"bis";T(2);"nm"
9555 PRINT "Bereich 2 von    ";T(12);"bis";T(13);"nm"
9557 PRINT "Normierung   von ";T(10);"bis";T(11);"nm"
9560 PRINT E1$;" Konz.";TAB(26);T(3);"nM"
9580 PRINT E2$;" Konz.";TAB(25);T(4);"uM"
9600 PRINT "Anzahl der Messpunkte";TAB(29);T(5)
9610 PRINT "Anzahl Repetitionen  ";TAB(30);T(8)
9640 PRINT "Spr 1: ";B(17);"ml  /  Spr 2: ";B(18);"ml"
9645 IF D4$[3,3]="1" THEN PRINT "Spritze 3  angeschl.
     ";B(19);"ml"
9650 PRINT "Delta Epsilon 1 :  ";T(17);"1/mM cm"
9655 PRINT "Delta Epsilon 2 :  ";T(18);"1/mM cm"
9657 PRINT "Assay Volumen : ";TAB(25);INT(1000*B(18)*
     B(7)/C4);"ul"
9660 RETURN

rem *******************************************************
rem **    MICHKIN software :  konzmsg.ksys      *********
rem ********    Author : Bruno Michel           ********
rem *******************************************************
rem
rem
rem
rem ************* 50 ! KONZMSG B.M. ******************
rem ****************** common data area **************
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],
     D2,D3$[20],D4$[20],B9$[8]
rem ************* global variables *****************
180 DIM A$[60],P9$[20],P8$[20],H1$[64],F$[1],F1$[1]
185 SHORT V(15)
195 INTEGER I,I5,I6
200 CLEAR
210 L(20)=1 @ P9$="Result      :     " @ P8$="1.st order
     constant:"
220 W1=400 @ W2=800 @ T6=0 @ T7=20 @ T8=.5 @ T9=.5 @
H5=20 @   H6=10 @ L1=.5 @ L2=.5
230 V(1)=1 @ V(2)=0 @ V(3)=550 @ V(4)=0 @ V(5)=1 @
V(6)=0 @  V(7)=10
235 V(8)=2 @ V(9)=5 @ V(10)=0 @ V(11)=1 @ V(12)=0 @
V(13)=1   @ V(14)=0 @ V(15)=0
240 S3(70)=1 @ S3(71)=0 @ S3(72)=0 @ S3(73)=0 @
S3(74)=2 @                S3(75)=0
250 ON ERROR GOTO 320 @ PLOTTER 1 @ IF C(8)=1 THEN 490
320 GOTO 800
490 IF C(15)#0 THEN 500
491 A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700
492 FOR I=1 TO 3 @ A$="*"&VAL$(I) @ GOSUB 7000 @
A$="*="&VAL$(W(I+5)) @ GOSUB 7000
495 NEXT I
rem ************ testmeasure menu *******************
rem
500 ON ERROR GOTO 500 @ STOP MEASURE
510 CLEAR @ DISP "******** Testmeasure  **********"
520 DISP "******** Main Menu    **********"
```

```
530 DISP "Please select a softkey: " @ DISP "TestM =
     Testmeasurement"
540 DISP "AutoM = Measure samples repeatedly"
550 DISP "ConcM = Measure Concentr. Menu "
570 DISP "Syrin = Manual moving of syringe"
580 DISP "Wash  = Washing of syringes"
590 DISP "Exit  = Emergency brake"
600 DISP "Refer.= Reference Measurement  (before first
     measurement)"
610 IF C(9)=0 THEN DISP "Baseline not yet measured"
rem ************** softkeys ***************************
650 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"ConcM" GOTO
     1000 @ ON KEY# 3,"Syrin" GOTO 860
655 ON KEY# 4,"Wash" GOTO 7500 @ ON KEY# 5,"MAINPG"
     GOTO 800 @ ON KEY# 6,"Refer." GOTO 750
660 ON KEY# 7,"AutoM" GOTO 9000 @ ON KEY# 8,"TestM"
     GOTO 4000 @ KEY LABEL @ GOTO 650
rem * routines performing actions of first set of
     softkeys *
750 GOSUB 1900 @ GOTO 500
800 ERASE STANDARD 1
805 CHAIN "Autost.KSYS"
810 CLEAR @ DISP "is host ready to receive data
     (Y/N)" @ INPUT H1$
820 IF H1$="Y" THEN V(15)=1 ELSE V(15)=0
830 GOTO 500
860 A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700
865 GOSUB 7250 @ GOTO 500
980 CLEAR @ DISP "No Reference Measured" @ BEEP @ WAIT
     2000
rem ********* Measure concentration menu *************
rem
1000 CLEAR @ DISP "**** Measure Concentr. Menu ****" @
     ON ERROR GOTO 500
```

```
1050 DISP "Range 1  from  ";T(1);" to";T(2);"nm" @ DISP
     "Range 2  from  ";T(12);" to";T(13);"nm"
1060 DISP "Int. ref. from ";T(10);" to";T(11);"nm"
1080 DISP "Epsilon value Range 1 and Range2" @ DISP
     "Substrate:";T(15);"and";T(16);"1/mMcm"
1085 DISP "Product :";T(15)+T(17);"and ";T(16)+T(18);
     "1/mMcm"
1090 DISP "Substrate: ";H5;@ DISP "  Enzyme: ";H6;" ul"
1100 DISP "Assay volume  : ";B(7);"ml"
1150 DISP "Spectrum from ";W1;" to ";W2
1170 DISP "Fill    Syr. 1  Syr.2  Syr. 3 "
1180 DISP TAB(9);C(1);TAB(16);C(2);TAB(23);
     C(3);TAB(29);"ml"
1190 ON ERROR GOTO 1000
rem ********** softkeys ******************************
1200 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 5,"AssayV"
     GOTO 1300 @ ON KEY# 6,"Spectr" GOTO 1500
1230 ON KEY# 2,"SubstrM" GOTO 2000 @ ON KEY# 3,"ProdM"
     GOTO 2500 @ ON KEY# 4,"Epsilon" GOTO 1350
1260 ON KEY# 7,"[E,S]" GOTO 1400 @ ON KEY# 8,"-Rang" G
     OTO 1420 @ KEY LABEL @ GOTO 1200
rem ***** routines performing actions for softkeys ****
1300 CLEAR @ DISP "Enter the volume of Assays in ml" @
     INPUT H1
1310 IF H1<.2 OR H1>1.5 THEN BEEP @ DISP "not possible"
     @ WAIT 3000 @ GOTO 1300
1320 B(7)=H1 @ GOTO 1000
1350 CLEAR @ DISP "Please enter the extinction-
     coefficients for range 1"
1355 DISP T(1);"bis";T(2);"nm" @ DISP "first substrate
     then product :" @ INPUT T(15),T(16)
1360 CLEAR @ DISP "Please enter the extinction-
     coefficients for range 2"
```

```
1365 DISP T(12);"bis";T(13);"nm" @ DISP "first subst
     rate then product :" @ INPUT H1,H2
1370 T(17)=T(15)-H1 @ T(18)=T(16)-H2 @ GOTO 1000
     1400 CLEAR @ DISP "Enter the volume of substrate
     in ul" @ INPUT H5
1405 CLEAR @ DISP "Enter the volume of enzyme in ul" @
     INPUT H6
1410 IF H5<0 OR H6<0 OR H5>500 OR H6>1000 THEN 1400
1415 GOTO 1000
1420 CLEAR @ DISP "Enter wavelength range" @ INPUT
     W1,W2
1422 IF W1<190 OR W2<194 OR W1>816 OR W2>820 OR W1 MOD
2#0 OR W2 MOD 2#0 THEN 1420
1425 GOTO 1000
rem ************** measure spectrum *******************
1500 ! mESSUNG DES SPEKTRUMS
1510 ERASE STATUS
1520 CLEAR
1530 LAMBDA W1 TO W2
1540 Y-SCALE @ ON ERROR GOTO 1660
1550 DISP "New mixture (Y/N)";@ INPUT H1$@ IF H1$<>"Y"
     THEN GOTO 1580
1560 GOSUB 1900 @ WAIT 1000
1565 DISP "*** The assay is mixed   *******"
1570 W1(3)=H5/1000 @ W1(2)=H6/1000 @ W1(1)=B(7)-W1(3) @
     GOSUB 7100
1572 A$="*H" @ GOSUB 7000 @ WAIT 3000 @ A$="*H" @ GOSUB
     7000 @ WAIT 4000
1580 MODE 0,0
1585 MEASURE 5
1590 GOSUB 5400
rem *************** error recovery ********************
1600 GOTO 1000
```

```
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP "Correct
     error condition +CONT";ERRL
1670 PAUSE
1675 GOTO 500
rem ************ measurement of baeline **************
1900 DISP "*** Measure of baseline   *****"
1901 IF C(1)<1 THEN BEEP @ DISP "Syringe 1 ist empty!"
     @ BEEP @ WAIT 4000 @ GOTO 500
1902 A$="*1" @ M=1 @ GOSUB 7000 @ A$="*A1.0" @ GOSUB
     7000
1903 A$="*B" @ GOSUB 7000 @ WAIT 4000 @ A$="*B" @ GOSUB
     7000 @ WAIT 4000
1905 LAMBDA W1 TO W2
1906 ABSORBANCE
1910 MODE 0,1
1920 REFERENCE 10
1930 IF NMEAS=0 THEN GOTO 1930
1970 C(9)=1 @ RETURN
rem ************* measure substrate *****************
2000 PRINT @ PRINT @ PRINT "*************************
     ******" @ PRINT @ CLEAR
2010 PRINT "Concentration measured at the
     substratabsorption"
2020 PRINT "dilutionfactor 1:";B(7)/H5*1000
2047 PRINT "No.    Range 1     Range 2  "
2050 W1(3)=H5/1000 @ W1(2)=0 @ W1(1)=B(7)-W1(2)-W1(3) @
     GOSUB 7100
2070 A$="*H" @ GOSUB 7000 @ WAIT 4000
2120 FOR K=1 TO T(8)
2150 A$="*H" @ GOSUB 7000 @ WAIT 1500+B(8)+B(9)/2
2160 T5=.5 @ Q1=0 @ GOSUB 3000
2200 PRINT USING "DD,XXXX,SD.DDDDD,XXXX,SD.DDDDD" ;
     K,H1,H2
2220 S4(K)=H1 @ S5(K)=H2 @ NEXT K
```

```
2240 PRINT "-------------------------------" @ PRINT "
     average of ";T(8);" Measurements:"
2250 H1=0 @ H2=0
2260 FOR K=1 TO T(8) @ H1=H1+S4(K) @ H2=H2+S5(K) @ NEXT
     K @  H1=H1/T(8) @ H2=H2/T(8)
2280 PRINT USING "XXXXXX,SD.DDDD,XXXX,SD.DDDD" ; H1,H2
2300 PRINT E2$;" Concentration"
2310 PRINT "in the Substratesyringe :"
2320 E1=H1/T(15)*1000*B(7)/W1(3) @ E2=H2/T(16)*1000*
     B(7)/W1(3)
2380 PRINT USING "XXXX,DDDDDD.DD,XX,DDDDDD.DD,3A" ;
     E1,E2,"   uM"
2390 PRINT "-------------------------------"
2400 DISP @ DISP @ GOSUB 5400
2420 PRINT @ PRINT @ PRINT
2490 GOTO 1000
rem ************ measure product **********************
2500 PRINT @ PRINT @ PRINT "********************
     ***********"   @ PRINT @ CLEAR
2510 PRINT "Concentration measured at the
     productabsorption"
2530 PRINT "dilutionfactor 1:";B(7)/H5*1000
2550 PRINT "No.    Range 1     Range 2  "
2560 W1(3)=H5/1000 @ W1(2)=H6/1000 @ W1(1)=B(7)-W1(2)-
     W1(3) @ GOSUB 7100
2565 A$="*H" @ GOSUB 7000 @ WAIT 3000
2570 FOR K=1 TO T(8)
2585 A$="*H" @ GOSUB 7000 @ WAIT 1500+B(8)+B(9)/2
2590 T5=.5 @ Q1=0 @ GOSUB 3000
2600 PRINT USING "DD,XXXX,SD.DDDDD,XXXX,SD.DDDDD" ;
     K,H1,H2
2620 S4(K)=H1 @ S5(K)=H2 @ NEXT K
2640 PRINT "-------------------------------" @ PRINT "
     average of ";T(8);" measurements:"
```

```
2650 H1=0 @ H2=0
2660 FOR K=1 TO T(8) @ H1=H1+S4(K) @ H2=H2+S5(K) @ NEXT
     K @ H1=H1/T(8) @ H2=H2/T(8)
2680 PRINT USING "XXXXXX,SD.DDDD,XXXX,SD.DDDD" ; H1,H2
2700 PRINT E2$;" Concentration"
2710 PRINT "in the substratesyringe :"
2740 E1=H1/(T(15)+T(17))*1000*B(7)/W1(3) @
     E2=H2/(T(16)+T(18))*1000*B(7)/W1(3)
2770 PRINT USING "XXXX,DDDDDD.DD,XX,DDDDDD.DD,3A" ;
     E1,E2," uM"
2790 PRINT "--------------------------------"
2800 DISP @ DISP @ GOSUB 5400
2820 PRINT @ PRINT @ PRINT
2890 GOTO 1000
rem *********** wait until result stable *************
3000 LAMBDA W1 TO W2
3010 ABSORBANCE
3020 ON KEY# 1,"Exit" GOTO 1000 @ ON KEY# 2,"Result"
     GOTO 3280
3030 ON KEY# 3,"Spec" GOTO 3260 @ OFF KEY# 4
3040 OFF KEY# 5 @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
3100 MODE 0,1 ! LOOP BEGIN
3110 MEASURE T5
3115 IF NMEAS=0 THEN 3115
3120 H1=0 @ I1=0
3130 FOR I=T(1) TO T(2)
3140 H1=H1+VALUE(I) @ I1=I1+1
3150 NEXT I @ H1=H1/I1
3160 H2=0 @ I1=0
3170 FOR I=T(12) TO T(13)
3180 H2=H2+VALUE(I) @ I1=I1+1
3190 NEXT I @ H2=H2/I1
3200 KEY LABEL
3201 DISP "***  Concentration measure  ***"
```

```
3210 DISP "    Range 1    Range 2   "
3220 DISP USING "XXX,SZ.DDDDD,XXXX,SZ.DDDDD" ; H1,H2
3230 IF Q1=1 THEN GOTO 3250
3240 GOTO 3000
3250 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4
3255 RETURN
3260 MODE 0,0 @ MEASURE 5
3265 IF NMEAS=0 THEN 3265
3270 PAUSE
3275 T5=.5 @ Q1=0 @ GOTO 3100
3280 T5=5 @ Q1=1 @ WAIT 1000
3290 OFF KEY# 2 @ OFF KEY# 3 @ GOTO 3100
rem *********** assayconditions **********************
4000 !
4010 CLEAR @ DISP "******** Testmeasure  **********" @
     F$="E"
4020 DISP "Function :      ";@ IF V(1)=1 THEN DISP
     "ABSORPTION" ELSE DISP "1.DERIVATIVE"
4030 DISP "Measure wavelength:";V(3);" nm"
4040 DISP "Y-Scale :";V(4);TAB(17);" to  ";V(5);" OD"
4050 DISP "Measure  :";V(6);TAB(17);" to  ";V(7);" sec"
4060 DISP "Tangent  :";V(8);TAB(17);" to  ";V(9);" sec"
4070 DISP "Mixture  :   buffer :";W1(1)*1000;" ul"
4080 DISP "Enzyme:";W1(2)*1000;"  Substrate:";W1(3)*
     1000;" ul"
4090 DISP "Delta OD/sec:";V(10);"Fact.";V(11)
4100 DISP P9$;TAB(22);V(11)*V(10)
4105 DISP P8$;TAB(22);V(13)*V(12)
4110 DISP "Fill    Syr. 1  Syr.2  Syr. 3 "
4120 DISP TAB(9);C(1);TAB(16);C(2);TAB(23);C(3);
     TAB(29);"ml"
4190 ON ERROR GOTO 500
rem ************** softkeys ***************************
4200 ON KEY# 1,"Exit" GOTO 500
```

```
4205 ON KEY# 2,"Mix" GOTO 4400
4210 ON KEY# 3,"Meas" GOTO 5000
4215 ON KEY# 4,"Ch.Sk" GOTO 4300
4220 ON KEY# 5,"Plot" GOTO 4450
4230 ON KEY# 6,"Funct" GOTO 4500
4240 ON KEY# 7,"
     -Msg" GOTO 4550
4250 ON KEY# 8,"Y-Sca" GOTO 4600
4270 KEY LABEL @ GOTO 4200
4300 ON KEY# 1,"Print" GOTO 4850
4305 ON KEY# 2,"Time" GOTO 4650
4310 ON KEY# 3,"Graph" GOTO 5250
4315 ON KEY# 4,"Ch.Sk" GOTO 4200
4320 ON KEY# 5,"PAxis" GOTO 4480
4330 ON KEY# 6,"Analy" GOTO 4700
4340 ON KEY# 7,"Fact." GOTO 4750
4350 ON KEY# 8,"Store" GOTO 6000
4370 KEY LABEL @ GOTO 4300
4390 GOTO 4390
rem ****** routines performing actions for softkeys ***
4400 CLEAR @ DISP "Please enter volume of buffer";@
     INPUT H1@ W1(1)=H1/1000
4405 IF W1(1)<0 OR W1(1)>2 THEN GOTO 4440
4410 DISP "Please enter volume of enzyme";@ INPUT H1@
     W1(2)=H1/1000
4415 IF W1(2)<0 OR W1(2)>1 THEN GOTO 4440
4420 DISP "Please enter volume of substrate";@ INPUT
     H1@ W1(3)=H1/1000
4425 IF W1(3)<0 OR W1(3)>.5 THEN GOTO 4440
4430 GOTO 4000
4440 DISP "Mixture not possible" @ BEEP @ WAIT 3000 @
     GOTO 4000
4450 CLEAR @ DISP "Please enter linetype (1-8)" @ INPUT
     H1@  GOSUB 4455 @ GOTO 4000
```

```
4455 ON ERROR GOTO 4470
4457 PLOTTER 705
4460 LINETYPE H1
4462 PLOTTER
4464 PLOTTER 1
4466 RETURN
4470 BEEP @ DISP "Plotter doesn't respond" @ ON ERROR
     GOTO 500
4475 RETURN
4480 PLOTTER 705
4485 AXIS
4490 PLOTTER 1
4495 GOTO 4000
4500 IF V(1)=1 THEN V(1)=2 ELSE V(1)=1
4510 GOTO 4000
4550 CLEAR @ DISP "Please enter measure wavelength";@
     INPUT H1
4555 IF H1<190 OR H1>820 OR H1 MOD 2#0 THEN 4550
4560 V(3)=H1 @ GOTO 4000
4600 CLEAR @ DISP "Please enter Y-Scale eg. 0,1
     (0,0=Auto Scale)" @ INPUT V(4),V(5)
4605 IF V(4)=V(5) THEN 4620
4610 OVERLAY 0,V(7),V(4),V(5)
4620 GOTO 4000
4650 CLEAR @ DISP "Please enter Measure time in sec
     (end only) eg. 10";@ INPUT V(7)
4655 L1=.5 @ L2=.5
4660 IF V(7)<5 THEN L1=.2 @ L2=.2
4665 IF V(7)>150 THEN L1=INT(V(7)/10)/10 @ L2=1
4670 IF V(4)=V(5) THEN 4695
4675 OVERLAY 0,V(7),V(4),V(5)
4695 GOTO 4000
4700 CLEAR @ DISP "Please enter start  and stop    Time
     for tangent";@ INPUT V(8),V(9)
```

```
4710 IF V(8)<0 OR V(9)<1 OR V(8)>V(7)-1 OR V(8)>V(7)
     THEN BEEP @ DISP "falsch" @ GOTO 4700
4720 GOSUB 4900 @ GOTO 4000
4750 CLEAR @ DISP "Please enter a factor";@ INPUT V(11)
4755 GOTO 4000
4800 !
4810 !
4850 CLEAR @ DISP "Please select Printout:
     1=Delta OD * factor"
4852 DISP "2=1st order constant            0=no
     printout"
4854 INPUT V(2)
4855 IF V(2)>2 OR V(2)<0 THEN V(2)=0
4860 IF V(2)=0 THEN 4870
4862 DISP "Please enter text to be printed"
4865 IF V(2)=1 THEN INPUT P9$
4866 IF V(2)=2 THEN INPUT P8$
4870 GOTO 4000
4900 H1=0 @ V(10)=0 @ V(12)=0 @ ON ERROR GOTO 4997
4910 FOR I=V(8) TO V(9) STEP L1
4915 IF V(1)=1 THEN H1=H1+VALUE(I+L1)-VALUE(I-L1)
4920 IF V(1)=2 THEN H1=H1+VALUE(I)
4925 NEXT I
4930 V(10)=INT(H1*100000/((V(9)-V(8))/L1+1))/100000
4940 H2=0 @ I1=INT(V(7)/2)
4950 H1=0 @ IF V(1)=2 THEN 4990
4960 FOR I=L1 TO I1-L1 STEP L1
4965 H3=VALUE(I-L1+I1)-VALUE(I-L1) @ H4=VALUE(I+L1+I1)-
     VALUE(I+L1)
4966 IF V(10)<0 THEN H3=-H3 @ H4=-H4
4970 IF H3>0 AND H4>0 THEN H3=LOG(H3) @ H4=LOG(H4) @
     H1=H1+(H4-H3)/L1 @ H2=H2+1
4975 NEXT I
4980 IF H2#0 THEN V(12)=H1/H2
```

```
4990 RETURN
4997 BEEP @ DISP "Error during Calculation" @ WAIT 2000
     @ ON ERROR GOTO 500 @ GOTO 4990
rem ********** serial assay measure program *******
rem
5000 CLEAR @ DISP "Please enter number of
     repetitions";@ INPUT R@ IF R=0 THEN 4000
5002 IF R>10 OR R<1 THEN 5000
5005 DISP "*** The assay is mixed   *******" @ K=0
5007 GOSUB 5010 @ GOTO 4000
5010 PLOTTER 1
5015 ABSORBANCE
5020 IF V(1)=1 THEN 5025
5021 MODE 0,1
5022 GOTO 5035
5025 IF V(5)#V(4) THEN 5035
5030 OVERLAY 0,V(7),-.01,1.5
5035 LAMBDA V(3)
5040 GOSUB 7100
5045 A$="*H" @ GOSUB 7000 @ WAIT 4000
5047 IF V(2)#0 THEN PRINT "Mix: P";W1(1)*1000;"
     E";W1(2)*1000;" S";W1(3)*1000;"ul"
5050 FOR I=1 TO R
5055 ERASE MEMORY I
5060 NEXT I
5065 FOR K1=1 TO R
5070 A$="*H" @ GOSUB 7000 @ WAIT 1500
5072 IF V(4)#V(5) THEN MODE 0,1 @ WAIT 1000
5075 MEASURE L1,L2,0,V(7)
5077 MODE 0,0
5080 IF NMEAS#V(7)/L1+1 THEN 5080
5081 TO MEMORY K1
5083 X=SPOLL(708) @ F1$="N" @ H1=BIT(X,3)
```

```
5084 IF F$="M" AND H1=1 AND K=S3(75) AND K1=R THEN H1=1
     ELSE H1=0
5085 IF H1=1 THEN A$="*"&VAL$(S3(74)) @ GOSUB 7000 @
     A$="*E8" @ GOSUB 7000 @ F1$="W"
5087 GOSUB 4900
5088 IF V(2)#0 THEN PRINT "Point:";K1;" Delta
     OD/sec:";V(10)
5090 NEXT K1 @ IF R=1 THEN 5130
5092 RECALL MEMORY 1
5095 TO STANDARD 1
5100 FOR I=2 TO R
5102 RECALL MEMORY I
5105 ABSORBANCE + STANDARD 1
5107 CALCULATE
5110 ERASE STANDARD 1
5115 TO STANDARD 1
5120 ABSORBANCE
5125 NEXT I
5129 RECALL STANDARD 1
5130 ABSORBANCE / R
5133 ERASE STANDARD 1
5134 CALCULATE
5135 TO STANDARD 1
5140 ABSORBANCE
5145 RECALL STANDARD 1
5147 ERASE STANDARD 1
5150 IF V(1)#2 THEN 5160
5155 DERIVATIVE
5160 IF V(4)#V(5) THEN 5175
5165 Y-SCALE
5170 PLOTTER
5175 GOSUB 4900
5200 PRINTER IS 2 @ IF R=1 THEN 5245
5210 IF V(2)#0 THEN PRINT "Average:"
```

```
5215 IF V(2)=1 THEN PRINT P9$;TAB(21);V(10)*V(11)
5240 IF V(2)=2 THEN PRINT P8$;TAB(21);V(12)*V(13)
5245 RETURN
5250 ! CALCULATE
5255 PLOTTER 1
5260 IF V(1)=2 THEN 5270
5265 ABSORBANCE
5267 GOTO 5280
5270 DERIVATIVE
5280 IF V(5)#V(4) THEN 5300
5285 Y-SCALE
5290 TIME SCALE 0 TO V(7)
5300 !
5310 PLOTTER
5320 GOSUB 4900
5350 !
5390 GOTO 4390
5400 ALPHA @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8 @ ON
     KEY# 2,"Forts" GOTO 5490
5402 ON KEY# 5,"Manual" GOTO 5500
5405 ON KEY# 3,"Graph" GOTO 5410 @ ON KEY# 4,"Alpha"
GOTO 5400 @ KEY LABEL @ GOTO 5420
5410 GRAPH
5420 GOTO 5420
5490 RETURN
5500 BEEP @ CLEAR @ DISP "Enter the manual commands now
     press (Cont) after having finished"
5510 PAUSE
5520 GOTO 5400
rem ***************store and send *******************
6000 ! Store / RS-232
6010 CLEAR @ DISP "Do you want to store the kinetics
     (0=No)";@ INPUT S3(71)
6020 IF S3(71)<0 THEN S3(71)=0
```

```
6030 CLEAR @ DISP "Do you want ot transmit (1/0)";@
     INPUT S3(72)
6040 IF S3(72)#1 THEN S3(72)=0
6042 DISP "comment" @ INPUT T1$
6045 GOSUB 6050 @ GOTO 4000
6050 IF S3(71)>0 THEN GOSUB 6100
6060 IF S3(72)=1 THEN GOSUB 6200
6070 RETURN
6100 !
6110 ON ERROR GOTO 6190
6120 H1=S3(71)
6130 A$=T1$[1,8] @ IF H1<0 OR H1>999 THEN 6190
6140 ERASE STANDARD 1
6150 TO STANDARD 1,A$
6160 RECALL STANDARD 1
6170 TO FILE H1
6180 ERASE STANDARD 1
6190 ON ERROR GOTO 500 @ RETURN
6200 ON ERROR GOTO 500
6210 CONTROL 10,2 ; 7
6220 CONTROL 10,3 ; 15
6230 CONTROL 10,4 ; 7
6240 CONTROL 10,5 ; 48
6250 CONTROL 10,11 ; 192
6260 CONTROL 10,14 ; 19
6270 CONTROL 10,15 ; 17
6280 SET TIMEOUT 10;3000
6290 ON TIMEOUT 10 GOTO 6400
6300 PRINTER IS 10
6310 PRINT "*****";T1$
6320 PRINTER
6330 PRINTER IS 2
6380 OFF TIMEOUT 10
6390 RETURN
```

```
6400 BEEP @ DISP "Host isn't ready to receive data"
6405 OFF TIMEOUT 10
6410 RETURN
rem ************* syringe control menu ****************
rem
6995 ! Motorsteuerung    ***
7000 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
7005 SET TIMEOUT 7;3000
7010 ON TIMEOUT 7 GOTO 7090
7015 A1$=A$[2,2]
7020 I5=SPOLL(708)
7025 IF BIT(I5,7) THEN WAIT 500 @ GOTO 7020
7030 IF BIT(I5,5) THEN DISP "OUT OF RANGE" @ F5$="2" @
     C(15)=0 @ GOTO 7082
7040 A$=A$&"," @ OUTPUT 708 ;A$
7052 IF A1$="1" THEN M=1
7054 IF A1$="2" THEN M=2
7056 IF A1$="3" THEN M=3
7060 IF A1$="B" OR A1$="D" THEN C(M)=C(M)-W(M+10)
7062 IF A1$="C" THEN C(M)=C(M)+W(M+10)
7063 IF A1$="F" THEN C(M)=0
7064 IF A1$="G" THEN C(M)=W(M)
7065 IF A1$="H" THEN C(1)=C(1)-W1(1) @ C(2)=C(2)-W1(2)
     @ C(3)=C(3)-W1(3)
7070 OFF TIMEOUT 7
7080 RETURN
7082 IF A1$="F" OR A1$="B" OR A1$="D" THEN 7080 ELSE
     7040
7090 DISP "Switch on ASSAYOMAT" @ BEEP @ WAIT 5000 @
     RESET 7 @ GOTO 7000
rem ************** emulate *K command *****************
7100 A$="*TF84A" @ I6=W1(4) @ GOSUB 7200
7110 I6=INT(W1(4)/(W1(1)*48000/W(1)))+257 @ GOSUB 7200
```

```
7115 IF W1(2)=0 THEN I6=60258 ELSE I6=INT(W1(4)/(W1(2)*
     48000/W(2)))+257
7120 GOSUB 7200 @ IF W1(3)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(3)*48000/W(3)))+257
7125 GOSUB 7200 @ ! DISP A$
7130 GOSUB 7000
7190 RETURN
7200 B$="     " @ H1$="0123456789ABCDEF"
7205 FOR J1=4 TO 1 STEP -1
7210 I=I6 MOD 16 @ B$[J1,J1]=H1$[I+1,I+1] @ I6=I6-I @
     I6=I6/16 @ NEXT J1
7215 A$=A$&"="&B$[3,4]&"="&B$[1,2]
7220 RETURN
rem ************** softkeys ***************************
7250 M=0 @ GOTO 7480
7280 ON KEY# 1,"Give" GOTO 7430 @ ON KEY# 2,"Suck" GOTO
     7420
7300 ON KEY# 3,"Back" GOTO 7400 @ ON KEY# 4,"Empty"
     GOTO 7460
7320 ON KEY# 5,VAL$(W(M+10)) GOTO 7360
7330 ON KEY# 6,VAL$(C(M)) GOTO 7440
7340 ON KEY# 7,"Retur" GOTO 7355
7350 ON KEY# 8,"Syr."&VAL$(M) GOTO 7480
7352 KEY LABEL @ GOTO 7280
7355 OFF KEY# 1 @ RETURN
7360 CLEAR @ DISP "Volume per Key";@ INPUT W(M+10)@
     A$="*A"&VAL$(W(M+10)) @ GOSUB 7000
7370 GOTO 7280
7400 A$="*D" @ GOSUB 7000 @ GOTO 7280
7420 A$="*C" @ GOSUB 7000 @ GOTO 7280
7430 A$="*B" @ GOSUB 7000 @ GOTO 7280
7440 A$="*G" @ GOSUB 7000 @ GOTO 7280
7460 A$="*F" @ GOSUB 7000 @ GOTO 7280
7480 M=M+1 @ IF M>3 THEN M=1
```

```
7485 A$="*"&VAL$(M) @ GOSUB 7000 @ GOTO 7280
rem **************** washing syringe *****************
7500 CLEAR @ DISP "No. of cycles and Syringe (1-3)";@
     INPUT H1,M
7510 IF M>3 OR M<1 OR H1<0 OR H1>99 THEN 7500
7520 A$="*"&VAL$(M) @ GOSUB 7000
7530 A$="*E"&VAL$(H1) @ GOSUB 7000
7535 IF H1 MOD 2=0 THEN C(M)=0 ELSE C(M)=W(M+5)
7540 RETURN
7545 GOSUB 7000 @ NEXT I1 @ GOTO 490
rem ************* read information from ASSAYOMATE ****
7700 SET TIMEOUT 7;2000 @ A$=""
7710 ON TIMEOUT 7 GOTO 7790
7720 ENTER 708 USING "#,B" ; H1@ A$=A$&CHR$(H1) @ IF
     H1=13 OR LEN(A$)>40 THEN 7740
7730 GOTO 7720
7740 OFF TIMEOUT 7 @ ON ERROR GOTO 7790
7750 C(1)=VAL(A$[10,16])
7760 C(2)=VAL(A$[20,26]) @ C(3)=VAL(A$[30,36]) @
C(15)=1
7790 ON ERROR GOTO 1600 @ RETURN
rem *************** spacing types ********************
7900 IF K=0 THEN N3=0 @ GOTO 7970
7910 ON T(19) GOTO 7920,7930,7950,7940,7960
7920 N3=4*L3^(K-1) @ GOTO 7970
7930 N3=K*300/T(5) @ IF K>T(5)/2 THEN N3=150+INT(K-
     T(5)/2)*600/T(5) @ GOTO 7970 ELSE GOTO 7970
7940 K1=T(5)+1-K @ N3=1/K1*600 @ GOTO 7970
7950 N3=5+K*100/T(5)+3*L3^(K-1) @ GOTO 7970
7960 N3=N3/L(20)
7970 W1(3)=N3*L(20)/10000
7980 W1(1)=B(7)-W1(2)-W1(3) @ IF I=9 THEN RETURN
7985 A$="*K"&VAL$(W1(1))&","&VAL$(W1(2))&",
"&VAL$(W1(3)) @ GOSUB 7100
```

```
7990 GOSUB 7000 @ RETURN
rem *************** calculate needed volumes **********
8200 I=0 ! CALC PULS
8210 GOSUB 7900 @ H2=B(3)
8220 H4=T(4)*W1(3)/B(7)
8230 H3=T1(K)*1000000/T(17)/H2 @ J1=T2(K)*1000000/
     T(18)/H2
8240 RETURN
rem ********** automatical measuring of samples *******
9000 !
9010 CLEAR @ DISP "******** Automeasure  **********"
9020 DISP "Function :      ";@ IF V(1)=1 THEN DISP
     "ABSORPTION" ELSE DISP "1.DERIVATIVE"
9030 DISP "Title of Job :" @ DISP T1$
9040 DISP "Number of Measure:";S3(70)
9050 DISP "Store     : ";@ IF S3(71)>0 THEN DISP
     "file";S3(71) ELSE DISP "off"
9060 DISP "Send      : ";@ IF S3(72)=1 THEN DISP "on"
     ELSE DISP "off"
9070 DISP "Plot      : ";@ IF S3(73)=1 THEN DISP "on"
     ELSE DISP "off"
9080 DISP "Wash Syringe:";S3(74)
9090 DISP "Fill     ";W(S3(74)+5);"ml"
9110 DISP "Fill    Syr. 1  Syr.2  Syr. 3 "
9120 DISP TAB(9);C(1);TAB(16);C(2);TAB(23);
     C(3);TAB(29);"ml"
9190 ON ERROR GOTO 500
rem ****************** softkeys *********************
9200 ON KEY# 1,"Exit" GOTO 500
9205 ON KEY# 2,"Job" GOTO 9400
9210 ON KEY# 3,"Meas" GOTO 9500
9215 ON KEY# 4,"Wash" GOTO 9600
9220 ON KEY# 5,"Title" GOTO 9650
9230 ON KEY# 6,"Number" GOTO 9700
```

```
9240 ON KEY# 7,"Store" GOTO 9750
9250 ON KEY# 8,"Plot" GOTO 9800
9270 KEY LABEL @ GOTO 9200
9400 CLEAR ! job
9410 FOR K=1 TO 69 @ S3(K)=0 @ NEXT K
9420 I1=1 @ I=1
9450 DISP "Please enter the";I;"th"
9455 DISP "number of rep., and the amount  of buffer,
     enzyme    and substrate in (ul)";
9460 INPUT S3(I1),S3(I1+1),S3(I1+2),S3(I1+3)
9465 IF S3(I1)>0 AND I1<51 THEN I1=I1+5 @ I=I+1 @ GOTO
     9450
9470 S3(75)=I-1
9490 GOTO 9000
rem ****************************************************
9500 CLEAR @ F$="M" @ F1$="N" !   messe
9501 ON KEY# 1 GOTO 9597
9502 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
9505 PRINTER IS 2 @ PRINT @ PRINT "Messung
     Nr.";S3(70);"  ";D3$ @ PRINT T1$ @ PRINT
9507 S3(70)=S3(70)+1 @ I=S3(74) @ V(2)=1
9510 DISP "Probe ";S3(70);" sucked in" @ BEEP @ DISP
     "Press key#9 at the"
9515 DISP "Assayomate, when the next probe is ready"
9520 IF F1$="W" THEN 9540
9525 X=SPOLL(708) @ IF BIT(X,3)=0 THEN BEEP @ DISP "No
     Probe ready" @ WAIT 3000 @ GOTO 9597
9530 A$="*"&VAL$(I) @ GOSUB 7000 @ A$="*="&VAL$(W(I+5))
     @ GOSUB 7000 @ A$="*E8" @ GOSUB 7000
9540 DISP "prepeare probe";S3(70)+1 @ DISP "within the
     next 2 minutes"
9550 FOR K=1 TO S3(75) @ I1=K*5-4
```

```
9560 R=S3(I1) @ W1(1)=S3(I1+1)/1000 @ W1(2)=S3(I1+2)/
     1000 @ W1(3)=S3(I1+3)/1000 @ GOSUB 5010
9570 GOSUB 6050
9580 IF S3(73)=1 THEN H1=(K-1) MOD 8 @ H1=H1+1 @ GOSUB
     4455
9585 NEXT K @ PRINT "--------------------------------"
     @ PRINT
9595 GOTO 9505
9597 PRINT @ PRINT @ PRINT @ PRINT @ GOTO 9000
rem ****************************************************
9600 ! wash
9610 CLEAR @ DISP "Please enter the syringe to be
     washed (2/3)";@ INPUT H1
9615 IF H1>3 OR H1<2 THEN 9610
9620 S3(74)=H1 @ I=H1
9630 DISP "Please enter the fill level";@ INPUT H1@ IF
     H1<0 OR H1>W(I) THEN 9630
9635 W(I+5)=H1 @ GOTO 9000
9650 CLEAR @ DISP "Please enter title of job:";@ FLIP @
     INPUT T1$@ FLIP
9670 IF LEN(T1$)<64 THEN T1$=T1$&" " @ GOTO 9670
9680 GOTO 9000
9700 ! nummer
9710 CLEAR @ DISP "Please enter the number of
     measurement";@ INPUT S3(70)@ GOTO 9000
9750 ! store
9760 CLEAR @ DISP "Do you want to store OD vs. time ?
     (0=No)";@ INPUT S3(71)
9770 IF S3(73)<0 THEN S3(73)=0
9780 CLEAR @ DISP "Do you want to send OD vs. Time
     (1/0)";@  INPUT S3(72)
9790 IF S3(72)#1 THEN S3(72)=0
9795 GOTO 9000
9800 IF S3(73)=0 THEN S3(73)=1 ELSE S3(73)=0
```

```
9810 GOTO 9000

rem **************************************************
rem *********    MICHKIN software :  asinit.ksys   ****
rem *********    Author : Bruno Michel             ****
rem **************************************************
rem
rem
rem
rem ****************** 1 ! ASINIT B.MICHEL ***********
rem
rem  This part of the MICHFIT software checks if the
rem  Assayomate is present and loads the valid setup
rem  down into the memory of the Assayomate.
rem
rem
rem ************ common data area *********************
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],
     D2,D3$[20],D4$[20],B9$[8]
rem **************** global variables *****************
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],
     F5$[1],A$[60],H1$[20]
195 INTEGER I,I2,I5,I6
200 F6$="0" @ Q1=1 @ Q2=1 @ F7$="0" @ I=0
210 F$="A" @ F5$="0" @ J=1 @ K=1 @ D4$[5,5]="1"
rem **** test if initialization has been performed ****
250 ON ERROR GOTO 900 @ IF C(8)#1 THEN GOTO 900
300 ON KEY# 1,"Exit" GOTO 900
310 DISP "Initialisation of Assayomat" @ DISP "key 1 =
     Exit"
490 IF C(15)#0 THEN 800
```

```
495 A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700
500 FOR I=1 TO 3
505 A$="*"&VAL$(I) @ GOSUB 7000
510 A$="*="&VAL$(W(I+5)) @ GOSUB 7000
520 A$="*X"&VAL$(W(I+15)) @ GOSUB 7000
530 A$="*A"&VAL$(W(I+10)) @ GOSUB 7000
540 ! A$="*A"&VAL$(W(I+10)) @ GOSUB 7000
590 NEXT I
800 RESET 7
810 IF D2=1 THEN D2=0 @ CHAIN "KIN 1  .KSYS"
820 IF D2=2 THEN D2=0 @ CHAIN "KIN 2  .KSYS"
830 IF D2=3 THEN D2=0 @ CHAIN "KONZMSG.KSYS"
900 RESET 7
910 CHAIN "Autost.KSYS"
rem ************ error recovery ***********************
1600 BEEP @ BEEP @ BEEP @ CLEAR @ DISP "Error
     Nr.";ERRN;"              ocurred on line ";ERRL
1610 DISP "correct errorcondition (+cont";ERRL;")"
1620 GOTO 900
rem **************** syringe control ******************
7000 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF     KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
7005 SET TIMEOUT 7;3000
7010 ON TIMEOUT 7 GOTO 7090
7015 A1$=A$[2,2]
7020 I5=SPOLL(708)
7025 IF BIT(I5,7) THEN WAIT 500 @ GOTO 7020
7030 IF BIT(I5,5) THEN DISP "Empty" @ F5$="2" @ C(15)=0
     @ GOTO 7082
7032 IF BIT(I5,4) THEN DISP "Full" @ C(15)=0
7035 IF BIT(I5,3) THEN C(15)=0
7040 A$=A$&"," @ OUTPUT 708 ;A$
7070 OFF TIMEOUT 7
7080 RETURN
```

```
7082 IF A1$="F" OR A1$="B" OR A1$="D" THEN 7080 ELSE
     7040
7090 DISP "Switch on ASSAYOMAT" @ BEEP @ WAIT 5000 @
     RESET 7   @ GOTO 7000
7100 DISP A$ @ A$="*TF84A " @ I6=W1(4) @ GOSUB 7200
7110 I6=INT(W1(4)/(W1(1)*48000/W(1)))+257 @ GOSUB 7200
7115 IF W1(2)=0 THEN I6=60258 ELSE I6=INT(W1(4)/(W1(2)
     *48000/W(2)))+257
7120 GOSUB 7200 @ IF W1(3)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(3)*48000/W(3)))+257
7125 GOSUB 7200
7130 GOSUB 7000
7190 RETURN
7200 B$="    " @ H1$="0123456789ABCDEF"
7205 FOR J1=4 TO 1 STEP -1
7210 I=I6 MOD 16 @ B$[J1,J1]=H1$[I+1,I+1] @ I6=I6-I @
     I6=I6/16 @ NEXT J1
7215 A$=A$&"="&B$[3,4]&"="&B$[1,2]
7220 RETURN
7700 SET TIMEOUT 7;2000 @ A$=""
7710 ON TIMEOUT 7 GOTO 7790
7720 ENTER 708 USING "#,B" ; H1@ A$=A$&CHR$(H1) @ IF
     H1=13 OR LEN(A$)>40 THEN 7740
7730 GOTO 7720
7740 OFF TIMEOUT 7 @ ON ERROR GOTO 7790
7750 C(1)=VAL(A$[10,16])
7760 C(2)=VAL(A$[20,26]) @ C(3)=VAL(A$[30,36]) @
     C(15)=1
7790 ON ERROR GOTO 1600 @ RETURN
```

APPENDIX E
MICHFIT Software Listings

```
rem  ************************************************
rem  *******   MICHFIT software :  ha.bas     *********
rem  *******   Author : Bruno Michel      *************
rem  ************************************************
rem idrives
rem  1 : screenfile+systemfile+directory 6: mainprogram
rem  2 : spoolfiles                      7 : plotprogram
rem  3 : ..........                 8 : regressionprogram
rem  4 : workfiles                      9 : inputprogram
rem  5 : plotfiles                      10 :
rem  ************************************************
rem
rem
rem
rem  ************************************************
rem                 Start of source code
rem  ************************************************
rem  ******************* definitions ****************
90 rem $include : 'comvar'
200 width 255 :
240 call graphoff : call cls : dim re(400),iquelle(20)
245 drives$="a:b:c:d:e:f:"
250 if pinit=1 then goto 1000
255 pinit=1 : tein=1 : tmod=1 : taus=1
260 anzgem=5: anztit=5 : anzspek=5 : forts=0 :
     pret$="ha "
265 for i=1 to 20 : pointer%(i)=0 : idrive(i)=1 :
     iquelle(i)=0 : next i : idrive(11)=2
270 for i=1 to 10 : tque(i)=i : sque(i)=i : next i
rem  **********get account and pasword ***************
```

```
290 col%=0 : row%=21 : call curs(col%,row%) : input
     "Please enter number of your account";Konto%
300 if Konto%>20 or Konto%<0 then goto 10000
310 close#1 : open "r",#1,"a:default.hsy",256 :
     field#1,256 as ex$
320 get #1,Konto%+55 : passw$=mid$(ex$,1,6) :
     name$=mid$(ex$,20,8) : initialen$=mid$(ex$,18,2)
330 call curs(col%,row%) : print space$(159) : call
     curs(col%,row%)
340 print "enter password "  : h1$=input$(6) : if
     h1$<>passw$ then goto 9999
rem ***************************************************
rem 350 close#8 : open "r",#8,file$,256 : field#8,256
     as ex$
rem 360 for i=1 to 2 : get#8,i+44
rem 370  for ii=1 to 127 : icon(i*127+ii-127+300)=cvi
     (mid$(ex$,ii*2-1,2))
rem 380  next ii
rem 390 next i : close#8 : ex$="" : call cls
rem **************initialize terminal *****************
400 print chr$(27);"&k0a0b0c0d1i0k0l0m0n0o0p1Q";
410 print chr$(27);"&s0a0b0c0d1g1h0j0k0l0m0n1W";
490 gosub 500 : gosub 600 : gosub 9000 : goto 1000
rem ************generate date string *****************
500 monate$="Jan.Feb.Mar.Apr.Mai JuniJuliAug.
     Sep.Okt.Nov.Dez."
510 tage$="01020304050607080910111213141516171819 20
     2122232425262728293031"
520 wtage$="Sonntag   Montag    Dienstag  Mittwoch
     DonnerstagFreitag   Samstag   "
530 call datum(jahr%,tag%,wtag%) : monat%=tag%/256 :
     tag%=tag% mod 256
```

```
540 call zeit(minuten%,sekunden%) : stunden%=minuten%
     /256 : minuten%=minuten% mod 256 : sekunden%=
     sekunden%/256
550 datst$="........  "+mid$(tage$,tag%*2-1,2)+"."+mid$
     (monate$,monat%*4-3,4)+" "+mid$(str$(jahr%),4,2)
560 call cls : print "Welcome to the MICHFIT software
     version Feb. 02. 1988"
570 print "Author : Bruno Michel" : print
580 print "todays date : ",datst$ : print "please
     insert your formatted data diskette into drive B:"
585 print "and wait for the completion of startup (10-
     30 sec)"
595 monate$="" : tage$="" : wtage$="" : return
600 rem *************initialize output module**********
rem sys$(1)=chr$(27)+"*pg0,3 -3,-5 6,0 -3,5Z"
rem sys$(2)=chr$(27)+"*pg0,-3 3,5 -5,0 3,-5Z"
rem sys$(3)=chr$(27)+"*pg-3,0 3,3 3,-3 -3,-3 -3,3Z"
rem sys$(4)=chr$(27)+"*pg-2,2 0,-4 4,0 0,4 -4,0Z"
rem sys$(5)=chr$(27)+"*pg-1,3 -2,-2 0,-2 2,-2 2,0 2,2
     0,2 -2,2 -2,0Z"
rem sys$(6)=chr$(27)+"*pg0,3 -3,-5 6,0 -3,5,0,-1,-2,-
     3,4,0,-2,3,0,-1,-1,-1,2,0,-1,1Z"
rem sys$(7)=chr$(27)+"*pg0,-3 3,5 -5,0 3,-5,0,1,-
     2,3,4,0,-2,-3,0,1,-1,1,2,0,-1,-1Z"
rem sys$(8)=chr$(27)+"*pg-3,0 3,3 3,-3 -3,-3 -3,3,1,
     0,2,2,2,-2,-2,-2,-2,2,1,0,1,1,1,-1,-1,-1,-1,1Z"
rem sys$(9)=chr$(27)+"*pg-2,2 0,-4 4,0 0,4 -4,0,1,-
     1,0,-2,2,0,0,2,-2,0Z"
rem sys$(10)=chr$(27)+"*pg-1,3 -2,-2 0,-2 2,-2 2,0 2,2
     0,2 -2,2 -2,0,1,0,-3,-3,3,-3,3,3,-3,3,0,-1,-2,-
     2,2,-2,2,2,-2,2,0,-1,-1,-1,1,-1,1,1,-1,1Z"
rem sym$(1)="UC -99,0,4,99,-3,-6,6,0,-3,6,-99;"
rem sym$(2)="UC -99,0,-4,99,3,6,-6,0,3,-6,-99;"
rem sym$(3)="UC -99,-3,0,99,3,3,3,-3,-3,-3,-3,3,-99;"
rem sym$(4)="UC -99,-2,2,99,0,-4,4,0,0,4,-4,0,-99;"
rem sym$(5)="UC -99,-1,3,99,-2,-2,0,-2,2,-2,2,0,2,2,0,
     2,-2,2,-2,0,-99;"
rem sym$(6)="UC -99,0,4,99,-3,-6,6,0,-3,6,0,-1,-2,-
     4,4,0,-2,4,0,-1,-1,-2,2,0,-1,-2,-99;"
rem sym$(7)="UC -99,0,-4,99,3,6,-6,0,3,-6,0,1,2,4,-
     4,0,2,-4,0,1,1,2,-2,0,1,-2,-99;"
rem sym$(8)="UC -99,-3,0,99,3,3,3,-3,-3,-3,-3,3,1,0,
     2,2,2,-2,-2,-2,-2,2,1,0,1,1,1,-1,-1,-1,-1,1,-99;"
rem sym$(9)="UC -99,-2,2,99,0,-4,4,0,0,4,-4,0,1,-1,0,-
     2,2,0,0,2,-2,0,-99;"
rem sym$(10)="UC -99,-1,3,99,-2,-2,0,-2,2,-2,2,0,2,2,0,
     2,-2,2,-2,0,0,-1,-1,-1,0,-2,1,-1,2,0,1,1,0,2,-
     1,1,-2,0,0,-1,0,-2,2,0,0,2,-2,0,-99;"
rem *********** default for plot definitions **********
800 for i=1 to 30 : tl(i)=0 : pga(i)=0 : next i
810 for i=1 to 20 : beschr$(i)=space$(80) : macro$(i)=
     space$(5) : next i
830 rem plx   ply   lrand   urand  vel  tickgr  linesp
840 pg(1)=20 : pg(2)=15 : pg(3)=3 : pg(4)=2 : pg(5)=20
     : pg(6)=1 : pg(7)=10
850 rem  pos1-4 Spooler.  stift  astift  lage
     annotate  annotstift
860 pg(8)=1 : pg(9)=1 : pg(10)=0 : pg(11)=2 : pg(12)=1
     : pg(13)=1 : pg(14)=0 : pg(15)=1
865 for i=16 to 20 : pg(i)=1 : pgx(i)=10  : pgy(i)=-
     5*(i-15) : next i
870 pxu#(1)=0 : pxu#(2)=10 : pxu#(3)=2 : pxu#(4)=0 :
     pxu#(5)=3
880 pxo#(1)=0 : pxo#(2)=10 : pxo#(3)=2 : pxo#(4)=0 :
     pxo#(5)=1
890 pyl#(1)=-0.01 : pyl#(2)=0.04 : pyl#(3)=0.01 :
     pyl#(4)=0 : pyl#(5)=3
```

```
900 pyr#(1)=-0.01 : pyr#(2)=0.04 : pyr#(3)=0.01 :
     pyr#(4)=0 : pyr#(5)=1
910 rem titel    Beschr.    Zahlen    Symbole    text
920 pgx(1)=.3:  pgx(2)=.25   :  pgx(3)=.20  :
     pgx(4)=.15   : pgx(5)=.20
930 pgy(1)=.4    : pgy(2)=.35  : pgy(3)=.25   :
     pgy(4)=.25  : pgy(5)=.25
940 rem pos tit    pos xubes    pos ylbes    pos xobes
     pos yrbes
945 pgx(6)=50   : pgx(7)=50   : pgx(8)=-10  : pgx(9)=50
     : pgx(10)=108
950 pgy(6)=108  : pgy(7)=-10  : pgy(8)=50    :
     pgy(9)=105  : pgy(10)=50
960 rem org xubes  pos ylbes  org xobes  org yrbes  ...
965 pgx(12)=-2   : pgx(13)=-5.5 : pgx(14)=-2   :
     pgx(15)=.5   : pgx(11)=1
970 pgy(12)=-1   : pgy(13)=-.25 : pgy(14)=.25  :
     pgy(15)=-.25 : pgy(11)=1
975 for i=1 to 10 : psym(i)=i : next i
980 for i=21 to 30 : pg(i)=0 : pgx(i)=0 : pgy(i)=0 :
     next i
990 return
995 rem ********* ende init ausgabemodul **************
rem ************** Main menu **************************
1000 call cls : call offtouch : rowinc%=0 : colinc%=10
1010 print "  MICHFIT software  Author : Bruno Michel
     Version  Feb. 01. 1988"
1020 PRINT "******************************************"
1030 print "*****                              *******"
1040 PRINT "*****      M A I N   M E N U       *******"
1042 print "*****                              *******"
1044 print "******************************************"
1045 col%=7 : row%=7 : call curs(col%,row%) : print
     "User name and fileidentification     : ";name$
```

```
1047 col%=7 : row%=8 : call curs(col%,row%) : print "
     Initials          :  ";initialen$
1048 print "                please select a touchfield or
     a softkey (f1-f8)"
1050 row%=13 : col%=1 : colinc%=15 : rowinc%=3 :
     resp$="A" : gosub 4000
1060 col%=2 : row%=14 : call curs(col%,row%) : print
     "Calculation of "
1065 row%=15 : call curs(col%,row%) :          print "
     Binding data "
1070 row%=13 : col%=21 : colinc%=15 : rowinc%=3 :
     resp$="B" : gosub 4000
1080 col%=22 : row%=14 : call curs(col%,row%) : print
     "Calculation of"
1085 row%=15 : call curs(col%,row%) :             print "
     Spectra "
1090 row%=13 : col%=41 : colinc%=15 : rowinc%=3 :
     resp$="C" : gosub 4000
1100 col%=42 : row%=14 : call curs(col%,row%) : print
     "Calculation of"
1105 row%=15 : call curs(col%,row%) :             print "
     Kinetic data"
1110 row%=13 : col%=61 : colinc%=15 : rowinc%=3 :
     resp$="D" : gosub 4000
1120 col%=62 : row%=14 : call curs(col%,row%) : print "
     Data        "
1125 row%=15 : call curs(col%,row%) :             print "
     Transfer   "
1130 h1$="1= Read   Diskette2=Initial.Spectra
     3=Initial.Kinetics4= Edit    Mask   5= New
     User   6= Edit   Sysfiles7=SpekctraReformat8= End
     Program "
1140 gosub 8700
rem ***************wait for input ********************
```

```
1400 print chr$(27); "-z2N";
1410 a$=input$(1): a=asc(a$)-17: b=asc(a$)-89
1420 print chr$(27); "-z0N";
1430 if a>0 and a<9 then 1700
1450 if a$="A" then gosub 4200 : i=1 : goto 1900
1455 if a$="B" then gosub 4300 : i=2 : goto 1900
1460 if a$="C" then gosub 4400 : i=3 : goto 1900
1465 if a$="D" then pret$="ha" : chain "dacom"
1655 goto 1400
rem ******** routines performing softkey functions ****
1700 on a goto 1720,1740,1760,1780,1800,1820,1840,1860
1720 call cls : gosub 9000 : goto 1000
1740 goto 2000
1760 goto 3000
1780 gosub 4500 : goto 1000
1800 goto 290
1820 goto 6000
1840 goto 5000
1860 goto 9999
rem ************** read systemfiles *****************
1900 col%=0 : row%=18 : call curs(col%,row%)
1901 i1=idrive(1) : h2$=mid$(drives$,i1*2-1,2) :
     close#1 : open "r",#1,h2$+name$+ext$,256 :
     field#1,256 as ex$
1902 get#1,1 : if mid$(ex$,3,2)="1=" then print
     "Systemfiles found on drive ";h2$ : goto 1940 else
     h4$=h2$
1903 i1=1 : h2$=mid$(drives$,i1*2-1,2) : h3$=name$ :
     close#1 : open "r",#1,h2$+name$+ext$,256 :
     field#1,256 as ex$
1904 get#1,1 : if mid$(ex$,3,2)="1=" then print
     "Systemfiles are copied to drive ";h2$ : gosub
     4100 : goto 1940
```

```
1905 h3$="DEFAULT" : close#1 : open "r",#1,h2$+h3$+
     ext$,256 : field#1,256 as ex$
1906 get#1,1 : if mid$(ex$,3,2)="1=" then print
     "Default Systemfiles are copied to ";h2$;name$ :
     gosub 4100 : goto 1940
1910 print "File :DEFAULT";ext$;" is missing" : goto
     1400
1940 rem
1960 rem
1970 h3$="" : h4$="" : h5$="" : m$="" : h1$="" :
     ext$="" : sext$=""
1980 ex$="" : ex2$="" : n$="" : for i=1 to 90 :
     format$(i)="" : next i : for i=1 to 20 :
     filename$(i)="" : next i
1990 h2$=mid$(drives$,idrive(6)*2-1,2) : x=fre("") :
     forts=0 : chain h2$+fort$
rem  **********Initialize spectra workfile ************
2000 rem
2020 col%=0 : row%=20 : call curs(col%,row%) : input
     "Are you sure to erase all spectra (Y/N) ";f$ : if
     f$<>"Y" then goto 1000
2050 for i=1 to 10 : ivar(i,1)=0 : nvar(i,1)=0 : next i
     : close#1 : open "r",#1,name$+".SVA",70 :
     n$=space$(70)
2610 for i=1 to 8 : mid$(n$,i*2-1,2)=mki$(ivar(i,1)) :
     next i : for i=1 to 6 : mid$(n$,i*4+13,4)=
     mks$(nvar(i,1)) : next i
2630 mid$(n$,41,30)="*............................." :
     field#1,70 as sva$ : lset sva$=n$
2650 for i=1 to 141 : put#1,i : next i : close #1 :
     n$=space$(130)
2730 open "r",#1,name$+".SPE",1300  : n$=space$(256) :
     for i=1 to 63 : mid$(n$,i*4-3,4)=mks$(0) : next i
```

```
2750 field#1,256 as e1$,256 as e2$,256 as e3$,256 as
     e4$,256 as e5$,20 as e6$
2760 for i=1 to 5 : gosub 2910 : next i : lset
     e6$=mid$(n$,1,20)
2770 for i=1 to 141 : put#1,i : next i : close #1 :
     goto 1000
2910 if dnr=1 then lset e1$=mid$(n$,1,252)
2920 if dnr=2 then lset e2$=mid$(n$,1,252)
2930 if dnr=3 then lset e3$=mid$(n$,1,252)
2940 if dnr=4 then lset e4$=mid$(n$,1,252)
2950 if dnr=5 then lset e5$=mid$(n$,1,252)
2960 return
rem **********Initialize kinetic workfile *************
3000 rem
3020 col%=0 : row%=20 : call curs(col%,row%) : input
     "Are you sure to erase all binding and kinetic
     data (Y/N)";f$ : if f$<>"Y" then goto 1000
3350 for i=1 to 10 : nvar(i,1)=0 : ivar(i,1)=0 : next i
     : h1=0 : n$=space$(130)
3400 for i=1 to 10 : mid$(n$,i*2-1,2)=mki$(ivar(i,1)) :
     mid$(n$,i*4+17,4)=mks$(nvar(i,1)) : next i
3520 mid$(n$,61,70)="*........xx*.........
     *..................*..................N"
3700 close#1 : open "r",#1,name$+".KVA",130 :
     field#1,130 as vak$ : lset vak$=n$
3720 for i=1 to 100 : put#1,i : next i : close #1 :
     open "r",#1,name$+".KIW",512 : n$=space$(256)
3740 for i=1 to 63 : mid$(n$,i*4-3,4)=mks$(h1) : next i
     : field#1,256 as e1$,256 as e2$
3760 for i=1 to 2 : gosub 2910 : next i
3770 for i=1 to 100 : put#1,i : next i : close #1 :
     goto 1000
rem ****************************************************
```

```
4000 call fntouch(row%,col%,rowinc%,colinc%,resp$) :
     return
4100 close#1 : open "r",#1,h2$+h3$+sext$,1760 :
     field#1,1760 as x$ : Print "copying ";h3$
4120 for i=1 to 15 : get#1,i : lset screen$=x$ :
     put#14,i : next i : close#1 : : x$=""
4130 close#1 : open "r",#1,h2$+h3$+ext$,256 :
     field#1,256 as x$
4140 close#2 : open "r",#2,h4$+name$+ext$,256 :
     field#2,256 as ex$
4150 for i=1 to 35 : get#1,i : lset ex$=x$ : put#2,i :
     next i
4160 h1$=space$(256) : for i=1 to 200 :
     mid$(h1$,i,1)="1" : next i : lset ex$=h1$ :
     put#2,36
4170 close#1 : close#2 : x$="" : ex$="" : h1$="" :
     return
4200 rem *********Bindung*****************************
4210 close #8 : open "r",#8,mid$(drives$,idrive(4)*2-
     1,2)+name$+".KVA",130 : field#8,130 as vat$
4220 close #9 : open "r",#9,mid$(drives$,idrive(4)*2-
     1,2)+name$+".KIW",512 : field#9,256 as e1$,256 as
     e2$
4230 close #10 : open "r",#10,mid$(drives$,idrive(1)*2-
     1,2)+name$+".DIR",8 : field#10,8 as direntry$
4240 close#14 : open "r",#14,mid$(drives$,idrive(1)*2-
     1,2)+name$+".BSC",1760 : field#14,1760 as screen$
4250 e3$="" : e4$="" : e5$="" : e6$="" : ext$=".BSY" :
     sext$=".BSC" : fort$="vbi"
4260 return
4300 rem
*************spektren**********************************
4310 close #8 : open "r",#8,mid$(drives$,idrive(4)*2-
     1,2)+name$+".SVA",70 : field#8,70 as vat$
```

```
4320 close #9 : open "r",#9,mid$(drives$,idrive(4)*2-
     1,2)+name$+".SPE",1300 : field#9,256 as e1$,256 as
     e2$,256 as e3$,256 as e4$,256 as e5$,20 as e6$
4330 close #10 : open "r",#10,mid$(drives$,idrive(1)*2-
     1,2)+name$+".DIR",8 : field#10,8 as direntry$
4340 close#14 : open "r",#14,mid$(drives$,idrive(1)*2-
     1,2)+name$+".SSC",1760 : field#14,1760 as screen$
4350 ext$=".SSY" : sext$=".SSC" : fort$="vsp"
4360 return
4400 rem *************kinetik*************************
4410 close #8 : open "r",#8,mid$(drives$,idrive(4)*2-
     1,2)+name$+".KVA",130 : field#8,130 as vat$
4420 close #9 : open "r",#9,mid$(drives$,idrive(4)*2-
     1,2)+name$+".KIW",512 : field#9,256 as e1$,256 as
     e2$
4430 close #10 : open "r",#10,mid$(drives$,idrive(1)*2-
     1,2)+name$+".DIR",8 : field#10,8 as direntry$
4440 close#14 : open "r",#14,mid$(drives$,idrive(1)*2-
     1,2)+name$+".KSC",1760 : field#14,1760 as screen$
4450 e3$="" : e4$="" : e5$="" : e6$="" : ext$=".KSY" :
     sext$=".KSC" : fort$="vki"
4460 return
rem ****************Mask editor ***********************
4500 h1$="1=Recall  Screen# 2=                3= Read
     Defaults4= Store  Screen #5= other      applikat6=
.................7=................8= End      Edit   "
4510 call offtouch : gosub 8700 : rem keylabel
4520 a%=0 : call cls : col%=0 : row%=22 : call
     curs(col%,row%) : print "Please select
     applikation: 1=Binding 2=Spectra 3=Kinetics" :
     input i
4530 h2$="BSK" : finame$=screendr$+name$ +"."+mid$(h2$,
     i,1)+"SC" : ext$="."+mid$(h2$,i,1)+"SC"
```

```
4540 a%=0 : call cls : col%=0 : row%=22 : call
     curs(col%,row%) : print "Please select softkey or
     edit Screen of ";finame$
4545 close#1 : open "r",#1,finame$,1760 : field#1,1760
     as x$
4550 while a%<>25 : call echo(a%)
4560 if a%=18 then gosub 4800 : rem read screen
4570 if a%=20 then gosub 4850 : rem read default screen
4580 if a%=21 then gosub 4900 : rem store screen
4585 if a%=22 then goto 4500
4590 wend
4600 close#1 :  a$=inkey$ : return
rem ****************************************************
4800 ii%=0 : def seg : x=varptr(#1) : if x<-32768 then
     ii%=x+65536 else ii%=x
4810 call curs(col%,row%) : print space$(79) : call
     curs(col%,row%) : input "Please enter screen # to
     read";i1 : get#1,i1
4820 call cls : call recscr(ii%) : call curs(col%,row%)
     : print "Please select softkey or edit Screen"
4830 return
4850 close#1 : open "r",#1,screendr$+"DEFAULT"+
     ext$,1760 : field#1,1760 as x$
4860 def seg : x=varptr(#1) : if x<-32768 then
     i=x+65536 else i=x
4870 call curs(col%,row%) : print space$(79) : call
     curs(col%,row%) : input "Please enter default
     screen # to read";i1 : get#1,i1
4880 call cls : call recscr(i) : call curs(col%,row%) :
     print "Please select softkey or edit Screen"
4890 close#1 : open "r",#1,finame$,1760 : field#1,1760
     as x$
4895 return
```

```
4900 def seg : x=varptr(#1) : if x<-32768 then
     i=x+65536 else i=x
4910 call curs(col%,row%) : print space$(79) : call
     curs(col%,row%) : input "Please enter screen # to
     store";i1 : call stoscr(i)
4920 put#1,i1 : call curs(col%,row%) : print space$(79)
     : call curs(col%,row%) : print "Please select
     softkey or edit Screen"
4930 return
5000 rem ******** Datensatz umschreiben*************
5050 call cls : close#8 : open "r",#8,name$+".SVA",70 :
     field#8,70 as vat$
5110 close#9 : open "r",#9,name$+".SPE",1300  :
     field#9,256 as e1$,256 as e2$,256 as e3$,256 as
     e4$,256 as e5$,20 as e6$
5200 for zr%=1 to 100 : gosub 9600 : gosub 9300
5220 nvar(3,zr%)=ivar(1,zr%) : nvar(4,zr%)=ivar(2,zr%)
5230 nvar(5,zr%)=2 : nvar(6,zr%)=0 : i1=ivar(3,zr%) :
     i2=ivar(4,zr%)
5240 ivar(6,zr%)=int((nvar(4,zr%)-nvar(3,zr%))/2+1)
5250 ivar(3,zr%)=int((i1-nvar(3,zr%))/2+1)
5260 ivar(4,zr%)=int((i2-nvar(3,zr%))/2+1)
5270 ivar(5,zr%)=0 : ivar(7,zr%)=0 : ivar(8,zr%)=0 : if
     ivar(3,zr%)<1 or ivar(4,zr%)<1 then goto 5350
5280 for i=1 to 8 : print ivar(i,zr%)"  "; : next i :
     print " "
5290 for i=1 to 6 : print nvar(i,zr%)"  "; : next i :
     print " "
5295 print mid$(vat$,41,30) : hi$=mid$(vat$,41,30)+"  "
5300 for i=1 to ivar(6,zr%)
5310 i1=(ivar(1,zr%)-180)/2+1
5320 re(i)=re(i+i1) : print re(i);"  ";
5330 next i : print "  " : print "ok? j" : a$=input$(1)
     : if a$<>"j" then goto 5420
```

```
5340 for i=ivar(6,zr%)+1 to 321 : re(i)=0 : next i :
     goto 5400
5350 for i=1 to 10 : ivar(i,zr%)=0 : nvar(i,zr%)=0 :
     next i
5355 for i=1 to 321 : re(i)=0 : next i : print " " :
     print zr%;" erased"
5360 mid$(hi$,1,30)="*............................."
5400 gosub 9510 : gosub 9400
5410 next zr%
5420 close#8 : close#9 : goto 1000
6000 rem **********edit systemprompts *****************
6010 call cls : print "Enter file (1=vsp  2=vki 3=vbi
     4=reg 5=ha)"
6030 input h1 : ifiletyp=h1 : pointer%(26)=0 :
     pointer%(25)=0
6040 if h1=1 then file$=name$+".ssy"
6041 if h1=2 then file$=name$+".ksy"
6042 if h1=3 then file$=name$+".bsy"
6044 if h1=4 then file$=name$+".rsy"
6045 if h1=5 then file$=name$+".hsy" : goto 7000
6060 close#8 : open "r",#8,file$,256 : field#8,256 as
     ex$
6210 for i=1 to 30 : get#8,i
6220  for ii=1 to 3 : l=cvi(mid$(ex$,ii*84-83,2))
6230   format$(i*3+ii-3)=mid$(ex$,ii*84-81,l)
6240  next ii
6250 next i
6300 for i=1 to 2 : get#8,i+30
6310  for ii=1 to 127 : icon(i*127+ii-127)=cvi(mid$
     (ex$,ii*2-1,2))
6320  next ii
6330 next i
6340 for i=1 to 3 : get#8,i+32
```

```
6350  for ii=1 to 63 : re(i*63+ii-63)=cvs(mid$(ex$,ii
     *4-3,4))
6360  next ii
6370 next i
6400 input "Please select page";seite : if seite>14 or
     seite<1 then goto 6400
6410 call cls
6450 for i=1 to 10
6454 if seite=10 then for ii=1 to 10 : col%=(ii-1)*7 :
     row%=i*2-2 : call curs(col%,row%) : print using
     "#######";icon((i-1)*10+ii) : next ii : goto 6470
6455 if seite=11 then for ii=1 to 10 : col%=(ii-1)*7 :
     row%=i*2-2 : call curs(col%,row%) : print using
     "#######";icon((i-1)*10+ii+100) : next ii : goto
     6470
6456 if seite=12 then for ii=1 to 10 : col%=(ii-1)*7 :
     row%=i*2-2 : call curs(col%,row%) : print using
     "#######";icon((i-1)*10+ii+200) : next ii : goto
     6470
6458 if seite=13 then for ii=1 to 10 : col%=(ii-1)*7 :
     row%=i*2-2 : call curs(col%,row%) : print using
     "####.##";re((i-1)*10+ii) : next ii : goto 6470
6459 if seite=14 then for ii=1 to 10 : col%=(ii-1)*7 :
     row%=i*2-2 : call curs(col%,row%) : print using
     "####.##";re((i-1)*10+ii+100) : next ii : goto
     6470
6460 col%=0 : row%=i*2-2 : call curs(col%,row%) : print
     format$(i+10*(seite-1))
6470 next i
6500 col%=0 : row%=21 : call curs(col%,row%)  : print
     "format page";seite;"  Please select line or
     999=exit 888=store 777 =new page"
6510 input h1 : if h1=999 then goto 1000
6520 if h1=888 then goto 6600
```

```
6525 if h1=777 then goto 6400
6527 if seite=10 or seite=11 or seite=12 then call
     curs(col%,row%) : print space$(79) : call
     curs(col%,row%) : print "former";icon(h1);" new";
     : input icon(h1) : goto 6500
6528 if seite=13 or seite=14 then call curs(col%,row%)
     : print space$(79) : call curs(col%,row%) : print
     "former";re(h1);" new";  : input re(h1) : goto
     6500
6530 if h1>14 or h1<1 then goto 6500
6540 col%=0 : row%=h1*2-2 : in$=format$(h1+10*(seite-
     1)) : leer=70 : gosub 30000
6545 format$(h1+10*(seite-1))=in$ : goto 6410
6550 goto 6500
6600 for i=1 to 30 : h1$=space$(256) : print "store";i
6620  for ii=1 to 3 : l=len(format$(i*3+ii-3)) :
     mid$(h1$,ii*84-83,2)=mki$(l)
6630   mid$(h1$,ii*84-81,l)=format$(i*3+ii-3)
6640  next ii : lset ex$=h1$ : put #8,i
6650 next i
6700 for i=1 to 2 : h1$=space$(256) : print
     "store";i+30
6710  for ii=1 to 127 : mid$(h1$,ii*2-
     1,2)=mki$(icon(i*127+ii-127))
6720  next ii : lset ex$=h1$ : put#8,i+30
6730 next i
6800 for i=1 to 3 : h1$=space$(256) : print
     "store";i+32
6810  for ii=1 to 63 : mid$(h1$,ii*4-
     3,4)=mks$(re(i*63+ii-63))
6820  next ii : lset ex$=h1$ : put#8,i+32
6830 next i
6840 close#8 : print "file closed"
6850 goto 1000
```

```
7000 rem *****edit basic systemfile ******************
7050 close#8 : open "r",#8,file$,256 : field#8,256 as
     ex$
7090 h1$=space$(256) : mid$(h1$,1,2)=mki$(0)
7200 for i=1 to 50 : get#8,i : l=cvi(mid$(ex$,1,2)) :
     if l<1 or l>255 then l=0
7230   format$(i)=mid$(ex$,3,l)
7250 next i
7300 for i=1 to 4 : get#8,i+50
7310  for ii=1 to 63 : re(i*63+ii-63)=cvs(mid$(ex$,ii
     *4-3,4))
7320  next ii
7330 next i
7340 for i=1 to 2 : get#8,i+54
7350  for ii=1 to 127 : icon(i*127+ii-127+300)=cvi(mid$
     (ex$,ii*2-1,2))
7360  next ii
7370 next i
7400 input "Please select page";seite : if seite>8 or
     seite<1 then goto 7400
7410 call cls
7450 for i=1 to 10
7455 if seite=6 or seite=7 then for ii=1 to 10 :
     col%=(ii-1)*7 : row%=i*2-2 : call curs(col%,row%)
     : print using "###.###";re((i-1)*10+ii+(seite-
     6)*100) : next ii
7460 if seite=8 or seite=9 then for ii=1 to 10 :
     col%=(ii-1)*7 : row%=i*2-2 : call curs(col%,row%)
     : print using "#####";icon((i-1)*10+ii+(seite-
     8)*100+300) : next ii
7465 if seite<6 then col%=0 : row%=i*2-2 : call
     curs(col%,row%) : print format$(i+10*(seite-1))
7470 next i
```

```
7500 col%=0 : row%=21 : call curs(col%,row%)  : print
     "page";seite;"  select line or      999=exit
     888=store 777 =new page 666=copy"
7510 input h1 : if h1=999 then goto 1000
7520 if h1=888 then goto 7600
7525 if h1=777 then goto 7400
7526 if h1=666 then call curs(col%,row%) : print
     space$(79) : call curs(col%,row%) : input "Quelle
     Ziel";h3,h4 : if h3<50 and h4<50 and h3>0 and h4>0
     then format$(h4)=format$(h3) : goto 7410 else goto
     7410
7527 if seite=8 or seite=9 then call curs(col%,row%) :
     print space$(79) : call curs(col%,row%) : print
     "former";icon(h1);" new";  : input icon(h1+300) :
     goto 7500
7528 if seite=6 or seite=7 then call curs(col%,row%) :
     print space$(79) : call curs(col%,row%) : print
     "former";re(h1);" new";  : input re(h1) : goto
     7500
7530 if h1>10 or h1<1 then goto 7500
7540 col%=0 : row%=h1*2-2 : in$=format$(h1+10*(seite-
     1)) : leer=70 : gosub 30000
7545 format$(h1+10*(seite-1))=in$ : goto 7410
7550 goto 7500
7600 print "File";file$;" is stored "
7610 for i=1 to 50 : h1$=space$(256)
7620  l=len(format$(i)) : mid$(h1$,1,2)=mki$(l)
7630  mid$(h1$,3,l)=format$(i)
7640  lset ex$=h1$ : put #8,i
7650 next i
7700 for i=1 to 4 : h1$=space$(256)
7710  for ii=1 to 63 : mid$(h1$,ii*4-3,4)=mks$(re(i
     *64+ii-64))
7720  next ii : lset ex$=h1$ : put#8,i+50
```

```
7730 next i
7750 for i=1 to 2 : h1$=space$(256)
7760  for ii=1 to 127 : mid$(h1$,ii*2-1,2)=mki$(icon
     (i*127+ii-127+300))
7770  next ii : lset ex$=h1$ : put#8,i+54
7780 next i
7790 close#8 : print "file closed"
7800 goto 1000
rem **************uppercase***************************
7900 il=len(in$)
7910 for i=1 to il : ia=asc(mid$(in$,i,1)) : if ia>96
     and ia<123 then ia=ia-32
7920 mid$(in$,i,1)=chr$(ia) : next i : return
rem ***************Keylabel**************************
8700 if len(h1$)<160 then h1$=h1$+space$(161-len(h1$))
8705 for i=1 to 8 : m$=str$(i) : h2$=chr$(27)+"&f0a"+
     mid$(m$,2,1)+"k16d1L"+mid$(h1$,i*18-15,16)+chr$
     (17+i) : print h2$; : next i
8710 print chr$(27); "&jB"; : return
rem *****************************************************
8800 ii=(idr-1)*5+ift : inhp(ii)=inhp(ii)+1 : if
     inhp(ii)>128 then inhp(ii)=128
8810 ii2=int((inhp(ii)-1)/32) : ii1=inhp(ii)-ii2*32 :
     ii2=ii2+(ift-1)*4+1 : if ii2=0 or ii1=0 then
     return
8820 mid$(filename$(ii2),ii1*8-7,8)=mid$(h1$,1,8) :
     return
rem ************error reading drive ******************
8900 resume 8910
8910 print "Error reading drive ";h3$;" a=abort r=retry
     ";err;" ";erl : a$=input$(1)
8920 if a$="a" then 9190
8930 if a$="r" then 9010
```

```
8950 print "Drive ";h3$;" contains no file : a=abort
     r=retry " : a$=input$(1) : goto 8920
rem *read directories and store in random access file *
9000 h5$="a:b:c:" : h2$="XSDKP"  : in$=initialen$ :
     gosub 7900
9005 for idr=1 to 2 : a%=0 : h3$=mid$(h5$,idr*2-1,2) :
     i1=0 : h1$=""
9010   on error goto 8900 : close#1 : open
     "o",#1,h3$+"system" : close#1
9015   print h3$ : pfad$=h3$+"*.*": i1=0 : finame$="" :
     x=fre("")
9020   for i=1 to 20 : filename$(i)=space$(256) : next
     i : for i=1 to 5 : inhp((idr-1)*5+i)=0 : next i
9025   finame$="                "+"" : def seg
9050   while a%=0
9055     if i1=1 then 9065
9060     i1=1 : call suchf(a%,pfad$,finame$) : if a%=18
     then 8950 else 9070
9065     finame$="              "+" " : call
     suchn(a%,finame$)
9070     h1$=finame$+space$(12)
9075     if idr=1 and mid$(h1$,1,4)="PLOT" then print
     h1$ : i=asc(mid$(h1$,5,1))-64 : print
     "spoolfile";i; " read in" : if i>pointer%(1) and
     i<20 then pointer%(1)=i
9125     if in$<>mid$(h1$,11,2) then 9150
9130     for ift=1 to 5
9135       if mid$(h1$,10,1)=mid$(h2$,ift,1) then gosub
8800 : print "*";h1$
9145     next ift
9150   wend
9155 close#1 : open "r",#1,name$+".DIR",8 : field#1,8
     as direntry$
9160 ii=(idr-1)*5*128+1 : ii2=1 : ii1=1
```

```
9165 for i=ii to ii+5*128 : iil=iil+1 : if iil=33 then
     iil=1 : ii2=ii2+1
9170 h1$=mid$(filename$(ii2),iil*8-7,8) : lset
     direntry$=h1$ : put#1,i
9175 next i : close#1 : for i=1 to 20 : filename$(i)=""
     : next i : direntry$=""
9190 next idr : x=fre("") : on error goto 0 : return
9300 rem *** read re () from spectra workfile *********
9305 if zr%<1 or zr%>141 then zr%=1
9310 get #9,zr% : i1=0 : i2=0 : dnr=1 : n$=e1$
9320 for wl=1 to 320 : i1=i1+1 : i2=i2+1
9330  re(i1)=cvs(mid$(n$,i2*4-3,4))
9340  if i2=63 then dnr=dnr+1 : gosub 9360 : i2=0
9350 next wl : return
9360 if dnr=2 then n$=e2$
9365 if dnr=3 then n$=e3$
9370 if dnr=4 then n$=e4$
9375 if dnr=5 then n$=e5$
9380 if dnr=6 then n$=e6$
9390 return
9400 rem ***** store spectra to workfile *************
9402 if zr%<1 or zr%>141 then zr%=1
9410 i1=0 : i2=0 : nvar(1,zr%)=9999 : nvar(2,zr%)=-9999
     : dnr=1
9415 m$=space$(252)
9420 for wl=1 to 320 : i1=i1+1 : i2=i2+1
9425  mid$(m$,i2*4-3,4)=mks$(re(i1))
9430  if i2=63 then gosub 9490 : dnr=dnr+1 : i2=0
9445  if re(i1)<>0 then if nvar(2,zr%)<re(i1) then
     nvar(2,zr%)=re(i1)
9446  if re(i1)<>0 then if nvar(1,zr%)>re(i1) then
     nvar(1,zr%)=re(i1)
9450 next wl : lset e6$=mid$(m$,1,20)
9470 put #9,zr% : gosub 9500
```

```
9480 return
9490 if dnr=1 then lset e1$=mid$(m$,1,252)
9491 if dnr=2 then lset e2$=mid$(m$,1,252)
9492 if dnr=3 then lset e3$=mid$(m$,1,252)
9493 if dnr=4 then lset e4$=mid$(m$,1,252)
9494 if dnr=5 then lset e5$=mid$(m$,1,252)
9498 return
9500 rem ****** store variables to workfile ***********
9502 if zr%<1 or zr%>141 then zr%=1
9505 get #8,zr% : hi$=mid$(vat$,41,30) :
     hi$=hi$+space$(30)
9510 n$=space$(70) : if nvar(2,zr%)>999 then
     nvar(2,zr%)=1
9511 if nvar(1,zr%)<-999 then nvar(1,zr%)=0
9515 for wl=1 to 8 : mid$(n$,wl*2-1,2)=mki$(ivar
     (wl,zr%)) : next wl
9520 for wl=1 to 6 : mid$(n$,wl*4+13,4)=mks$
     (nvar(wl,zr%)) : next wl
9535 mid$(n$,41,30)=hi$
9540 lset vat$=n$ : put #8,zr%
9550 return
9600 rem ******** read variables from workfile ********
9605 if zr%<1 or zr%>141 then zr%=1
9610 get #8,zr%
9620 for wl=1 to 8 : ivar(wl,zr%)=cvi(mid$(vat$,wl*2-
     1,2)) : next wl
9630 for wl=1 to 6 : nvar(wl,zr%)=cvs(mid$(vat$,wl
     *4+13,4)) : next wl
9660 return
9999 end
10000 rem *******Create new accounts *****************
10010 col%=0 : row%=21 : call curs(col%,row%) : print
      space$(159) : call curs(col%,row%)
```

```
10020 print "Geben Sie das Eroeffnungspasswort" :
     h1$=input$(6) : if h1$<>"axolot" then goto 9999
10022 close#1 : open "r",#1,"default.hsy",256 :
     field#1,256 as ex$
10025 for i=1 to 20 : get#1,i+55 : row%=i : call
     curs(col%,row%) : print i,mid$(ex$,20,8) : next i
10030 row%=21 : call curs(col%,row%) : print
     space$(159) : call curs(col%,row%) : input "Geben
     Sie die Kontonummer";Konto% : if Konto%>20 or
     Konto%<0 then goto 10030
10040 call curs(col%,row%) : print space$(159) : call
     curs(col%,row%) : input "Geben Sie den Namen (8
     Zeichen) und die Initialen (2 Zeichen)";name
     $,initialen$ : if len(name$)>8 or len(initialen
     $)<>2 then goto 10040
10050 call curs(col%,row%) : print space$(159) : call
     curs(col%,row%) : input "Geben Sie Ihr
     persoenliches Passwort (6 Zeichen)";passw$ :
     h1$=space$(256) : if len(passw$)<>6 then goto
     10050
10060 mid$(h1$,20,8)=name$ : mid$(h1$,18,2)=initialen$
     : mid$(h1$,1,6)=passw$
10070 lset ex$=h1$ : put#1,Konto%+55
10080 close#2 : open "r",#2,"a:"+name$+".hsy",256 :
     field#2,256 as ex2$
10090 for i=1 to 54 : get#1,i : lset ex2$=ex$ : put#2,i
     : next i : close#2
10100 goto 9999
30000 rem *********input routine ********************
30010 ip=1 : ins=0 : on error goto 30900 : con%=73 :
     rox%=row%
30020 call curs(col%,row%) : print in$;space$(ileer) :
     call curs(col%,row%) : ilmax=len(in$)
```

```
30100 com%=col%+ip-1 : if com%>80 then com%=com%-80 :
     rox%=row%+1 else rox%=row%
30102 call curs(com%,row%) : a$=inkey$ : if len(a$)=1
     then goto 30120
30105 if len(a$)=2 then goto 30120
30110 goto 30100
30120 if a$=chr$(27) then ins=1 : a$=" " else ins=0
30130 if a$=chr$(127) then goto 30500
30200 ia=asc(a$)
30210 if ia=13 then goto 30300
30220 if ia=9 then if ip<ilmax+1 then ip=ip+1 :
     com%=col%+ip-1 : goto 30100 else goto 30100
30230 if ia=8 then if ip>1 then ip=ip-1 : com%=col%+ip-
     1 : goto 30100 else goto 30100
30250 if ip<=ilmax and ins=0 then mid$(in$,ip,1)=a$ :
     print a$ : ip=ip+1 : goto 30100
30260 if ip<=ilmax and ins=1 then in$=in$+" " : for
     ii=len(in$)-1 to ip step-1 : mid$(in$,ii+1,1)
     =mid$(in$,ii,1) : next ii : mid$(in$,ip,1)=a$ :
     goto 30020
30270 in$=in$+a$ : ilmax=ip : ip=ip+1 : print a$ : goto
     30100
30300 if in$="exit" or in$="EXIT" or in$="STOP" or
     in$="nein" or in$="stop" then inerr=1 : goto 40000
30310 if inz=0 then goto 30400
30320 eing=val(in$)
30400 return
30500 if ip>len(in$) then goto 30100
30510 for ii=ip to len(in$)-1 : mid$(in$,ii,1)
     =mid$(in$,ii+1,1) : next ii
30520 h1$=in$ : l=len(in$)-1 : in$=mid$(h1$,1,l) : goto
30020
30900 print "input error" : goto 1000
40000 print "end" : end
```

```
rem  **************************************************
rem  *****     MICHFIT software :  comvar.bas       ***
rem  ******     Author : Bruno Michel          *********
rem  **************************************************
rem
rem
rem
rem
rem
rem ***************************************************
rem            definition of global variables
rem ***************************************************
rem
rem
rem option base 1   start counting from 1 for array
     numbering
rem defint i,j,w,t  all variables starting with i,j,w,t
     are
rem             integer
rem dim format$(90),
rem  sym$(10),
rem  sys$(10),
rem  inhalt$(10),
rem  beschr$(20),
rem  macro$(20),
rem  filename$(20)
rem dim icon(450),
rem  idrive(20)
rem dim psym(10),
rem  dmax(10),
rem  stift%(10),
rem  tl(30),
rem  tque(10),
rem  sque(20),
rem  inhp(10),
rem  c(10)
rem dim dx(10,332),
rem  dy(10,332),
rem  pg(30),
rem  pxu#(5),
rem  pyl#(5),
rem  pxo#(5),
rem  pyr#(5),
rem  pgx(30),
rem  pgy(30),
rem  pga(30)
rem dim ivar(10,100),
rem  nvar(10,100),
rem  idat(10),
rem  idx(128),
rem  idy(128),
rem  idp(128),
rem  pointer%(30),
rem  iu(100),
rem  ip(100)
rem
rem
rem
rem ***************************************************
rem                common variables
rem ***************************************************
rem
rem
rem
rem common pinit,
rem  forts,
rem  tmod,
rem  taus,
rem  tein,teart,idat()
rem common name$,
rem  initialen$,
rem  pret$,
rem  datst$
rem common anzgem,
rem  anztit,
rem  anzspek,
rem  autoscale,
rem  interpolate,
rem  anzeige,
rem  default
rem common softk,
rem  anzart,
rem  offo%,
rem  offi%
rem common format$(),
ren  sym$(),
rem  sys$(),
rem  inhalt$(),
rem  beschr$(),
rem  macro$(),
rem  filename$()
rem common icon(),
rem  idrive()
rem common psym(),
rem  dmax(),
rem  stift%(),
rem  tl(),
rem  tque(),
rem  sque(),
rem  inhp(),
rem  c(10)
rem common dx(),
rem  dy(),
rem  pg(),
rem  pxu#(),
rem  pyl#(),
rem  pxo#(),
rem  pyr#(),
rem  pgx(),
rem  pgy(),
rem  pga()
rem common ivar(),
rem  nvar(),
rem  idx(),
rem  idy(),
rem  idp(),
rem  pointer%(),
rem  iu(),
rem  ip()
rem
rem
rem ***************************************************
rem                    start of source code
rem ***************************************************
rem
rem
rem *********** declaration of variables ************
100 option base 1
105 defint i,j,w,t
110 dim format$(90),sym$(10),sys$(10),inhalt$(10),
     beschr$(20),macro$(20),filename$(20)
112 dim icon(450),idrive(20)
115 dim psym(10),dmax(10),stift%(10),tl(30),tque(10),
     sque(20),inhp(10),c(10)
```

```
120 dim dx(10,332),dy(10,332),pg(30),pxu#(5),pyl#(5),
     pxo#(5),pyr#(5),pgx(30),pgy(30),pga(30)
125 dim ivar(10,100),nvar(10,100),idat(10),idx(128),
     idy(128),idp(128),pointer%(30),iu(100),ip(100)
rem
rem **************** common declarations *************
rem
140 common pinit,forts,tmod,taus,tein,teart,idat()
145 common name$,initialen$,pret$,datst$
150 common anzgem,anztit,anzspek,autoscale,interpolate,
     anzeige,default
152 common softk,anzart,offo%,offi%

160 common format$(),sym$(),sys$(),inhalt$(),beschr$(),
     macro$(),filename$()
165 common icon(),idrive()
170 common psym(),dmax(),stift%(),tl(),tque(),sque(),
     inhp(),c(10)
175 common dx(),dy(),pg(),pxu#(),pyl#(),pxo#(),pyr#(),
     pgx(),pgy(),pga()
180 common ivar(),nvar(),idx(),idy(),idp(),pointer%(),
     iu(),ip()
rem
rem
rem ***************************************************

rem  **************************************************
rem  *******   MICHFIT Software: :   vki.bas     ******
rem  *******   Author :    Bruno Michel        ********
rem  **************************************************
rem
rem list of variables and pointers :
rem
rem ***************************************************
rem  meaning of variables stored in workfile :
rem
rem  ivar 1 :  Measuring wavelength from     nvar 1 :
     Y-maximum of data
rem  ivar 2 :  Measuring wavelength to       nvar 2 :
     X-maximum of data
rem  ivar 3 :  Output window (from point #)  nvar 3 :
     Enzyme concentration
rem  ivar 4 :  Output window (to point #)    nvar 4 :
     1. parameter of fit
rem  ivar 5 :  No. of stored data points     nvar 5 :
     2. parameter of fit
rem  ivar 6 :  Output window (to point #)    nvar 6 :
     3. parameter of fit
rem  ivar 7 :  Normalizing wavelength from   nvar 7 :
     4. parameter of fit
rem  ivar 8 :  Normalizing wavelength to     nvar 8 :
     5. parameter of fit
rem  ivar 9 :                                nvar 9 :
     6. parameter of fit
rem *****************************************************
rem  data structure of workfile :
rem
rem  present :
rem  63 x points in condensed binary format
rem  63 y points in condensed binary format
rem  each data set is stored in two consecutive records
     of the workfile
rem  suggested :
rem  16 x/y points per record of 256 bytes stored in
     condensed binary format
rem  n times sucha a data block per data curve as
     defined in first random
rem  acces block of 256 bytes
rem  each data set is stored in n consecutive records
     of the workfile
rem  **************************************************
rem  system pointer%(ii) functions :
rem
rem  1 :  No. of plots in outputspool       16 :
rem  2 :  idmax (Max. No. of drawpoints)    17 :
rem  3 :  current macroline                 18 :
     current direct. page
rem  4 :  Interpreter On off                19 :
     current menu
rem  5 :  output in process                 20 :
     initialisation pointer
rem  6 :  position during string output
     rem  7 :
rem  8 :  current input page
rem  9 :
rem 10 :
rem 11 :  Input Menu update
rem 12 :  Output menu update
rem 13 :  Content menu update
rem 14 :  Plotsize menu update
rem 15 :
rem ***************************************************
rem  pointers and flags
rem
rem  pinit     initialization performed : this flag is
               set if the
rem            common data area has been initialized
rem  forts     proceed pointer : this pointer is used
               to activate the
rem            input, data handling or output menu when
               the data pro-
rem            cessing subroutine is activated
```

```
rem  tmod       active transformation (1-6)
rem  tein       active input device (1-6)
rem  taus       active output device (1-6)
rem  teart      file type
rem  idat
rem  name       name of user (used to identify workfile)
rem  initialen initials of user (used for data file
                extensions)
rem  pret       name of calling program to be
                reactivated after termi-
rem             nation of active subroutine
rem  datstr     string holding date
rem  anzgem     No. of input data files to average
rem  anztit     No. of data curves to be processed
                simultaneously (10)
rem  autoscale autoscale flag 1/0
rem  interpolate    interpolate flag 1/0
rem  anzeige    actual page of data handling menu
rem  softk      active set of softleys
rem  offo%      roll state of output menu
rem  offi%      roll state of input menu
rem  format ()      prompts in desired language read
                from system file
rem             during startup
rem  sym ()     user definable symbols read from system
                files during startup
rem  inhalt () content of active directory page
                (contains 32 filenames of the active
                filetype (e.g. kinetic files)
rem  inhp ()    pointer for number of entries of each
                filetype in each drive (5 filetypes 5
                drives)
rem  macro ()  macros for command language
rem  filename ()filenames are used to hand over file
                names to the input subroutine
rem  iu ()      screen update pointers; is used to store
                the parts of the menus that have to be
                updated if the computer is not busy
rem  ip ()      workfile update pointers; is used to
                store the records that have been changed
rem  **************************************************
rem  10 data curves
rem
rem  psym ()    symbols resp. line types (1-50)
rem  stift% () pen (0-8) for plotter and display
rem  tque ()    source registers in input and output
                menu
rem  dmax ()    pointers for the number of data points
                of each data
rem             curve (1-16*block size)
rem  dx (,)     array of data points
rem  dy (,)     array of data points
rem  **************************************************
rem  plot definitions :
rem
rem  beschr () annotate strings, title, and labels of
                axes
rem  tl ()      string lengths
rem  pg ()  pga ()  plot definitions (see plot
                subroutine pl.exe)
rem  pxu#() pxo#()  minimum, maximum, delta x and delta
                xo
rem  pyl#() pyr#()  minimum, maximum, delta y and delta
                yo
rem  pgx () pgy ()  sizes of title, labels, digits,
                symbols and annotate
rem  idx () idy ()  draw data points (integer x and y)
```

```
                rem  idp ()         pen, symbol or line
                type of resp. draw point
rem  ***************************************************
rem  (touch/mouse) field definitions :
rem
rem  icon ()   definitions of fields (see main routine
                ha.exe)
rem  ***************************************************
rem  start of source code :
rem  ************* definitions      *******************
90 rem $include : 'comvar'
220 width 255
230 dim p$(10),interpk(10)
240 dim re(500),rex(500),rey(500),zr(10),ik(10),ist(10)
245 xx=1: yy=1: zz=0 : smooth=2 : running%=1 : iplot=0
     : ilock=0 : repeat%=0 : einl%=0 : touchs=0
250 onoff$="  NO   YES  " : filekenn$="XSDKP" :
     exp$="+#.###^^^^" : drt$="a:b:c:d:e:f:"
251 insav$="Interpreter on   exit=" : pret$="vki" :
     if tein<3 then drive$=mid$(drt$,tein*2-1,2)
255 scrndr$=mid$(drt$,idrive(1)*2-1,2) :
     dirdr$=mid$(drt$,idrive(2)*2-1,2) :
     spooldr$=mid$(drt$,idrive(3)*2-1,2)
260 sysdr$=mid$(drt$,idrive(4)*2-1,2) : formdr$=mid
     $(drt$,idrive(5)*2-1,2)
265 plpgdr$=mid$(drt$,idrive(6)*2-1,2) : mapgdr$=mid
     $(drt$,idrive(7)*2-1,2) : repgdr$=mid$(drt$,idrive
     (8)*2-1,2)
275 inpgdr$=mid$(drt$,idrive(9)*2-1,2) : hapgdr$=mid$
     (drt$,idrive(10)*2-1,2)
280 plotfdr$=mid$(drt$,idrive(11)*2-1,2)
285 idmax=pointer%(2) : comfile%=pointer%(3) :
     interpreter=pointer%(4) : ime=pointer%(19)
290 if pinit<>1 then goto 850
```

```
300 close #6 : close #7 : field#8,130 as vat$ :
    field#9,256 as e1$,256 as e2$
310 field#10,8 as direntry$ : field#14,1760 as screen$
320 close #11 : open "LPT1:" for output as #11 len=80
rem  *************  start depending on fort  *********
330 if forts=0 then gosub 4100 : gosub 1000 : ip(20)=1
340 if forts=1 then gosub 1000
350 if forts=2 then gosub 2000
360 if forts=3 then gosub 3000
370 if forts=4 then gosub 4000 : gosub 2000
380 if forts=5 then gosub 18500 : gosub 1000
390 if forts=6 then gosub 18500 : gosub 2000
400 if forts=7 then gosub 18500 : gosub 3000
470 if forts=8 then gosub 8000 : gosub 2000
480 if forts>8 then gosub 4100 : gosub 1000
rem  **************************************************
rem                  Main program
rem
******************************************************
500 while running%
505 if warnz%>0 then gosub 6300
510 if comfile%>0 then a$=macro$(comfile%) : comfile%=
    comfile%+1 : if comfile%>21 then comfile%=0
515 if comfile%>0 then gosub 4500
520 if comfile%=0 and interpreter=1 then gosub 4500
550 tshare=10 : ileer=1 : ilock=0 : if comfile%=0 and
    interpreter<>1 then gosub 5000 :
    rem ********warte   auf Eingabe **
560 if ime<1 or ime>6 then ime=1
565 if iflag=0 then 700
570 on ime gosub 1400,2400,3400,6700,6950,17400
700 if pointer%(20)<100 then pointer%(20)=pointer%
    (20)+1 : zr%=pointer%(20) : gosub 9800
710 on ime gosub 1025,2025,3025,990,990,990
```

```
790 wend
800 gosub 900
830 if ifort=1 then pointer%(2)=idmax : pointer%
     (3)=comfile% : pointer%(4)=interpreter
840 if ifort=1 then pointer%(19)=ime : chain fort$
rem  ********  Terminate data handling routine  *******
850 close #6 : close #7 : close #8 : close #9 :
     close#11 : close#14
860 gosub 950 : call offtouch : call cls
890 print "Auswertungsprogramm beendet" : end
rem  ***********  clean string space   **************
900 h$="" : h1$="" : h2$="" : h3$="" : h4$="" : h5$=""
     : h6$="" : h9$="" : h1b$="" : hx$="" : hy$=""
910 in$="" : intp$="" : form$="" : warn$="" : ex$="" :
     prompt$="" : close#1,#2,#3,#4,#5,#6,#7
920 i=fre("") : return : rem garbage collection
rem  ***********   save current status   *********
950 close#1 : open "r",#1,name$+".KSY",256  :
     field#1,256 as ex$ : h1$=space$(256)
955  for ii=1 to 100 : if iu(ii)=0 then
     mid$(h1$,ii,1)="0" else mid$(h1$,ii,1)="1"
960  next ii
965  for ii=101 to 200 : if ip(ii-100)=0 then
     mid$(h1$,ii,1)="0" else mid$(h1$,ii,1)="1"
970  next ii : lset ex$=h1$ : put#1,36 : close#1 :
     print "save current status? (y/n)" : a$=input$(1)
     : if a$<>"y" then on error goto 988 : kill
     "a:"+name$+".KST" : goto 989
975 open "o",#1,"a:"+name$+".KST" : for i=1 to 10 :
     print#1,tque(i) : next i : for i=1 to 10 :
     print#1,psym(i),stift%(i) : next i
980 for i=1 to 40 : print#1,pointer%(i) : next i :
     print#1,tein,teart,anzgem,anztit,tmod,taus,autosca
     le,interpolate,anzeige,anzart,tein,teart,anzgem,an
     ztit,tmod,taus,interpolate,anzeige,anzart
982 print#1,format$(15) : print#1,format$(16)
985 pg(8)=0 : isav=anztit : anztit=0 : gosub 8035 :
     anztit=isav
986 return
988 resume 989
989 on error goto 6000 : goto 986
990 return
rem ****************************************************
rem                Subroutines  /  procedures
rem ****************************************************
rem  **************  Input menu    *****************
rem
1000 on error goto 0 : if ime=1 then 1020
1005 call offtouch : call alpha : def seg : get#14,1 :
      call cls : i1=varptr(#14) : call recscr(i1)
1010 ime=1 : gosub 8500 : h1$=format$(1)+format$(2) :
      i1=1 : gosub 8700 : rem  call keylabel
1020 ip(19)=1 : pointer%(11)=0 : return
1025 if pointer%(11)>20 then return
1030 pointer%(11)=pointer%(11)+1 : if ip(pointer%(11))
      =0 then return
1040 on pointer%(11) gosub 1200,1205,1220,1225,
      1250,1250,4200,1300,1300,1300,1300,1100,1100,1100,
      1100,1100,1100,1100,1390,6800
rem       1    2    3    4    5    6    7    8    9
10   11   12   13   14   15   16   17   18   19   20
1045 ip(pointer%(11))=0 : if pointer%(11)<19 then
      ip(20)=1
1047 return
1100 return
1150 while pointer%(11)<21 : gosub 1025 : wend : return
rem **********block 1**********************************
```

```
1200 col%=22 : row%=2 : call curs(col%,row%) : print
     mid$(format$(29),tein*10-9,10);"  " : return
1205 col%=64 : row%=2 : call curs(col%,row%) : print
     mid$(format$(30),teart*15-14,15);"  " : return
1220 col%=22 : row%=4 : call curs(col%,row%) : print "
     ";anzgem;"     " : return
1225 col%=64 : row%=4 : call curs(col%,row%) : print "
     ";anztit;"  " : return
1250 j=pointer%(11)-4 : row%=6+2*j : col%=19 :
     h1$=space$(62)
1280  for i=1 to 10 : j1=10*(j-1)+i : if j1<=anztit
     then mid$(h1$,i*6-3,3)=str$(tque(j1))+"   "
1285  next i : call curs(col%,row%) : print h1$ :
     return
1300 j=pointer%(11)-7 : row%=12+2*j : h1$=space$(80)
1305  for i=1 to 8  : mid$(h1$,i*10-9,8)=mid$(inhalt
     $(1),(j-1)*64+i*8-7,8)
1310  next i : col%=0 : call curs(col%,row%) : if j=4
     then mid$(h1$,62,16)=" top      next p"
1320  print h1$ : return
1390 row%=21 : col%=0 : call curs(col%,row%) : print
     space$(150) : return
rem ***** Subroutines for input menu commands *********
1400 on error goto 6000 : if tein<3 and tein>0 then
     drive$=mid$(drt$,tein*2-1,2)
1420 col%=0 : row%=22 : call curs(col%,row%)
1430 if tfeld>0 and tfeld<11 then gosub 1450
1435 if tarray>0 and tarray<3 then gosub 1600
1440 if tsoft>0 and tsoft<9 then gosub 1700
1445 return
rem ********** actions of single fields ***************
1450 pointer%(11)=0 : on tfeld gosub
1460,1470,1480,1490,1500,1510,1520,1530,1540,1550
1455 ip(19)=1 : return
```

```
1460 tein=tein+1 : if tein>4 then tein=1
1465 ip(1)=1 : ip(7)=1 : return
1470 teart=teart+1 : if teart>5 then teart=1
1475 ip(2)=1 : ip(7)=1 : return
1480 prompt$="?" : col%=23 : row%=4 : in$="__" : ispa=3
     : gosub 3960 : anzgem=h1 : ip(3)=1 : return
1490 col%=64 : row%=4 : prompt$="?" : in$="__" : ispa=3
     : gosub 3960 : anztit=h1 : ip(4)=1 : ip(5)=1 : :
     ip(6)=1 : gosub 1766 : return
1500 prompt$=format$(19) : in$="__" : ispa=2 : gosub
     3960 : tque(1)=h1 : if tque(1)<0 or tque(1)>100
     then goto 1500
1505 for i=2 to anztit : tque(i)=tque(1)+i-1 : next i :
     ip(5)=1 : return
1510 gosub 6800 : gosub 6500 : return
1520 gosub 6800 : gosub 6500 : return
1530 gosub 4350 : return
1540 return
1550 return
rem ********** actions of arrays  ********************
1600 if tarray=1 then gosub 1610 else gosub 1655
1605 return
1610 ip(20)=1 : i=tcol : j=trow : if trow>2 then goto
1650
1620 row%=6+j*2 : col%=i*6+14 : j1=(j-1)*10+i : ispa=3
     : ip(5)=1 : ip(6)=1 : gosub 1766
1630 prompt$="?" : in$=str$(tque(j1)) : gosub 3960 :
     tque(j1)=h1 : if tque(j1)<1 or tque(j1)>99 then
     1620
1650 return
1655 if trow=4 and tcol>6 then 1690 else print "please
     select softkey first" : ip(19)=1 : return
1660 i=tcol : j=trow : if trow=4 and tcol>6 then 1690
```

```
1670 row%=12+j*2 : col%=i*10-2 : call curs(col%,row%) :
     print "*" : touchf=(trow-1)*8+tcol
1680 return
1690 ip(7)=1 : pointer%(11)=0 : ip(20)=1     : if
     tcol=8 then pointer%(8)=pointer%(8)+1
1695 if tcol=7 then pointer%(8)=1
1697 return
rem ****** routines performing softkey functions ******
1700 on tsoft gosub
1720,1740,1750,1780,1800,1820,1840,1860
1710 ip(19)=1 : pointer%(11)=0 : return
1720 gosub 6800 : gosub 950 : pinit=1 : ifort=1 :
     fort$=mapgdr$+"ha" : ime=0 : running%=0 :
     pointer%(11)=21
1725 close#8 : close#9 : close#10 : close#14 : vat$=""
     : e1$="" : e2$="" : screen$=""
1730 for i=1 to 90 : format$(i)="" : next i : for i=1
     to 20 : filename$(i)="" : next i : return
1740 j=1 : on error goto 6000 : print format$(16)
1741 gosub 1900 : filename$(1)=h1$ : if i2=1 and i1=1
     then h5=touchf else h5=0
1743 gosub 1900 : if i2=1 and i1=1 then gosub 1975 : if
     h1=1 then 1743
1745 call curs(col%,row%) : print filename$(1);"
     renamed to ";h1$;space$(20)
1746 name filename$(1) as h1$ : if h5<>0 then touchf=h5
     : gosub 1950 : gosub 1880 : return
1750 gosub 6800 : forts=1 : inu=1 : if tein=4 then call
     cls : fort$=inpgdr$+"iki" : ifort=1 : running%=0 :
     return
1751 print format$(11) : on error goto 6000
1752 gosub 1900 : filename$(inu)=h1$ : inu=inu+1 : if
     anzgem>1 and inu<anzgem+1 and teart<>5 then 1752
1757 if teart=5 or inu>=anztit+1 then 1760 else 1752
```

```
1760 if teart>5 or teart<1 then return
1762 if teart=5 then 1770
1764 gosub 6800 : ime=0 : fort$=inpgdr$+"iki" : ifort=1
     : running%=0
1766 for i=27 to 50 : ip(i)=1 : next i : for i=90 to 99
     : ip(i)=0 : next i
1767 for i=1 to anztit : iu(tque(i))=1 : next i
1768 return
1770 rem
1778 col%=0 : row%=22 : call curs(col%,row%) : print
     space$(79) : j=1 : gosub 7520 : gosub 2000 :
     return
1780 print format$(12) : gosub 1900 : h2$=drive$+
     "rescue.sys"
1785 on error goto 1797 : kill h2$ : col%=0 : row%=22 :
     call curs(col%,row%) : print "File ";h1$;"
     geloescht";space$(20)
1787 on error goto 1798 : name h1$ as h2$
1790 on error goto 6000 : gosub 1950 : gosub 4200 :
     gosub 1880 : return
1797 resume 1787
1798 resume 1790
1800 on error goto 6000 : print format$(13) : gosub
     1900 : gosub 6800
1810 filename$(1)=h1$ : j=1 : call cls : close#1 : open
     "i",#1,filename$(1)
1812 print format$(14) : input#1,h1$ : print h1$
1814 while not eof(1) : input#1,h1$ : print h1$
1815 ilock=1 : gosub 5000 : if a$="E" or a$="e" or
     a$="a" then 1819
1816 if a$="p" then print#11,h1$ : goto 1814
1817 if a$="n" then goto 1814 else goto 1815
1818 wend
1819 ilock=0 : ime=0 : gosub 1000 : return
```

```
1820 j=1 : on error goto 6000 : print format$(15)
1821 gosub 1900 : filename$(1)=h1$
1822 gosub 1900 : if i1=1 and i2=1 then gosub 1975 : if
     h1=1 then print "file exists" : goto 1822
1823 call curs(col%,row%) : print filename$(1);" is
     copied to ";h1$ : close#1 : open "i",#1,filename$
     (1) : close#2 : open "o",#2,h1$
1824 while not eof(1) : input#1,h2$ : print#2,h2$
1826 wend : gosub 1885 : return
1840 on error goto 6000 : gosub 6800 : ime=0 : call cls
     : filename$(1)=drive$+"*.*" : files filename$(1) :
     print "hit any key" : a$=input$(1)
1850 gosub 1000 : return
1860 running%=0 : return
1880 pointer%(11)=0 : ip(7)=1 : return
1885 pointer%(11)=0 : ip(7)=1 : return
rem *********get string or touchfield*****************
1900 eing%=1 : gosub 5000 : if iflag=0 then 1900 else
     if a$="E" or a$="A" then print "stopped" : goto
     500
1901 if tarray=2 then 1905 else call curs(col%,row%) :
     print space$(79) : call curs(col%,row%)
1903 prompt$="enter filename ? " : in$="a:TEST    .XBM"
     : ileer=14 : gosub 40000 : gosub 7990 : if
     mid$(in$,1,1)="E" then  print "stopped" : goto 500
1904 h1$=mid$(in$,1,14) : goto 1910
1905 gosub 1660 : if pointer%(11)=0 then for ii=1 to 18
     : gosub 1025 : next ii : goto 1900
1907 h1$=drive$+mid$(inhalt$(1),touchf*8-
     7,8)+"."+mid$(filekenn$,teart,1)+initialen$
1910 i3=0 : i4=0 : for i=1 to len(h1$) : if
     mid$(h1$,i,1)=":" then i3=i
1920  if mid$(h1$,i,1)="." then i4=i
```

```
1925 next i : if i3<>2 or i4<>11 or mid$(h1$,i3+1,i4-
     i3)=space$(8) then call curs(col%,row%) : print
     "try again"  : goto 1900
1930 if mid$(h1$,1,2)=drive$ then i1=1 else i1=0
1935 if mid$(h1$,i4+1,3)=mid$(filekenn$,teart,1)+
     initialen$ then i2=1 else i2=0
1945 col%=0 : row%=22 : call curs(col%,row%) : print
     space$(79) : call curs(col%,row%) : return
rem ***************delete entry***********************
1950 i2=teart+(tein-1)*5 : if tarray<>2 then return
1960 for i=touchf to inhp(i2) : i1=i+(tein-
     1)*640+(teart-1)*128+(pointer%(8)-1)*24 :
     get#10,i1+1 : put#10,i1 : next i :
     inhp(i2)=inhp(i2)-1
1965 gosub 4200 : return
rem *************add entry ***************************
1975 h1$=h1$+space$(8) : h2$=mid$(h1$,i3+1,i4-i3) :
     i2=(tein-1)*5+teart
1977 for i=1 to inhp(i2) : i1=(tein-1)*640+(teart-
     1)*128+i : get#10,i1 : if direntry$=h2$ then print
     "file exists" : h1=1 else h1=0
1980 next i : i1=(tein-1)*640+(teart-1)*128+inhp(i2)
1985 lset direntry$=h2$ : put#10,i1 : inhp(i2)=inhp
     (i2)+1
1990 gosub 4200 : return
rem ****************  Output menu  *******************
rem
2000 on error goto 0 : if ime=2 then 2020
2002 call offtouch : call alpha : if anztit>10 then
     anztit=10
2005 def seg : get#14,2 : call cls : i1=varptr(#14) :
     call recscr(i1)
2010 ime=2 : gosub 8500 : h1$=format$(31)+format$(32) :
     i1=2 : gosub 8700 : rem  call keylabel
```

```
2020 ip(39)=1 : pointer%(12)=0 : return
2025 if pointer%(12)>30 then return
2030 pointer%(12)=pointer%(12)+1 : if
     ip(pointer%(12)+20)=0 then return
2035 if pointer%(12)>20 then 2050
2040 on pointer%(12) gosub
2200,2205,2210,2215,2220,2225,2230,2295,2295,2295,2295,
2295,2100,2100,2100,2100,2100,2100,2290,6800
rem           1    2    3    4    5    6    7    8    9
10   11   12   13   14   15   16   17   18   19   20
2045 ip(pointer%(12)+20)=0 : if pointer%(12)<19 then
     ip(40)=1
2047 return
2050 j=pointer%(12)-20 : gosub 2905 : ip(pointer%(12)
     +20)=0 : return
2100 return
rem ************block1*************************
2200 col%=27 : row%=0 : call curs(col%,row%) : print
     mid$(datst$,1,20);"  ** Benutzer : ";mid$(name$,
     1,8);"******" : return
2205 col%=22 : row%=2 : call curs(col%,row%) : print
     mid$(format$(27),tmod*13-12,13) : return
2210 col%=64 : row%=2 : call curs(col%,row%) : print
     mid$(format$(28),taus*10-9,10);"    " : return
2215 col%=22 : row%=4 : call curs(col%,row%) : print
     mid$(onoff$,autoscale*6+1,6) : return
2220 col%=64 : row%=4 : call curs(col%,row%) : print
     mid$(onoff$,default*6+1,6) : return
2225 col%=22 : row%=6 : call curs(col%,row%) : print
     mid$(onoff$,interpolate*6+1,6) : return
2230 col%=64 : row%=6 : call curs(col%,row%) : print "
     ";anztit;"  " : return
2290 row%=21 : col%=0 : call curs(col%,row%) : print
     space$(150) : return
```

```
rem *****************  single line  *****************
2295 j=pointer%(12)-7
2300 i=j+offo% : form$=format$(35) : col%=0 :
     row%=j*2+10 : call curs(col%,row%)
2310 if i>anztit then print space$(160) : goto 2390
2320 zr%=tque(i) : gosub 9800 : if i=1 then
     enzs$=mid$(vat$,111,19)
2330 h1=abs(nvar(3,tque(i))) : if h1<0.001 or h1>99
     then mid$(form$,35,12)="+.####^^^^  "
2340 h1=abs(nvar(1,tque(i))) : if h1<0.1 or h1>999 then
     mid$(form$,47,12)="+.####^^^^  "
2350 h1=abs(nvar(2,tque(i))) : if h1<0.1 or h1>999 then
     mid$(form$,59,12)="+.####^^^^  "
2360 print using form$;tque(i),mid$(vat$,61,17),psym
     (i),stift%(i),nvar(3,tque(i)),nvar(1,tque(i)),nvar
     (5,tque(i)),ivar(3,tque(i))-1,ivar(4,tque(i))-1
2370 row%=j*2+11 : call curs(col%,row%)
2380 if anzart=2 then print using format$(36);mid$
     (vat$,79,2),mid$(vat$,81,10),mid$(vat$,91,16),mid$
     (vat$,111,14),nvar(2,tque(i)),nvar(9,tque(i)),nvar
     (10,tque(i)),nvar(6,tque(i)),ivar(5,tque(i)) else
     print space$(79)
2390 return
rem ****** subroutines for output menu commands  ******
2400 row%=22 : col%=0 : call curs(col%,row%) : print
     space$(79) : pointer%(12)=0
2405 row%=0 : call curs(col%,row%) : row%=22 : call
     curs(col%,row%)
2410 if tfeld>0 and tfeld<16 then gosub 2437
2420 if tarray>0 and tarray<2 then gosub 2600
2430 a=tsoft : if a>0 and a<9 then gosub 2700
2435 return
rem *********  single fields  ***********************
```

```
2437 if tfeld<7 then col%=int(icon(99+tfeld*2)/100) :
     row%=icon(99+tfeld*2)-col%*100 : call
     curs(col%,row%)
2440 ileer=5 : on tfeld gosub
2450,2460,2470,2480,2490,2500,2510,2520,2530,2540,2550,
2560,2570,2580,2590
2445 ip(39)=1 : return
2450 tmod=tmod+1 : if tmod>6 or tmod<1 then tmod=1
2455 default=0 : ip(22)=1 : for i=41 to 50 : ip(i)=1 :
     next i : return
2460 taus=taus+1 : if taus>6 or taus<1 then taus=1
2465 ip(23)=1 : ip(62)=1 : return
2470 autoscale=autoscale+1 : if autoscale>1 then
     autoscale=0
2475 ip(24)=1 : return
2480 default=1 : gosub 18000 : ip(24)=0 : return
2490 interpolate=interpolate+1 : if interpolate>1 then
     interpolate=0
2495 ip(25)=1 : return
2500 prompt$="?" : in$="__" : ispa=2 : gosub 3960 : if
     h1>=0 or anztit<10 then anztit=h1 : gosub 2100
2505 for i=27 to 32 : ip(i)=1 : next i : for i=41 to 50
     : ip(i)=1 : next i : for i=4 to 6 : ip(i)=1 : next
     i : pointer%(11)=0 : return
2510 prompt$=format$(45) : in$="_" : ispa=2 : gosub
3960 : if h1<1 or h1>100-anztit then return
2515 for i=1 to anztit : ip(i+27)=1 : tque(i)=h1+i-1 :
     next i : for i=41 to 50 : ip(i)=1 : next i : for
     i=4 to 6 : ip(i)=1 : next i : pointer%(11)=0 :
     return
2520 prompt$=format$(46) : in$="_" : ispa=2 : gosub
3960 : if h1<1 or h1>50-anztit then return
2525 for i=1 to anztit : ip(i+27)=1 : psym(i)=h1+i-1 :
     next i : return
```

```
2530 prompt$=format$(47) : in$="_,_" : ispa=4 : gosub
3960 : if h1<0 or h1>8 then return
2532 pg(12)=h1
2535 if h2<0 or h2>8 then return
2537 for i=1 to anztit : ip(i+27)=1 : stift%(i)=h2 :
     next i : return
2540 prompt$=format$(48) : in$="_" : ispa=7 : gosub
     3960
2545 for i=1 to anztit : ip(i+27)=1 : ip(i+40)=1 :
     nvar(3,tque(i))=h1 : zr%=tque(i) : gosub 9900 :
     next i : return
2550 prompt$=format$(49) : in$="_" : ispa=7 : gosub
     3960
2555 for i=1 to anztit : ip(i+27)=1 : ip(i+40)=1 :
     nvar(1,tque(i))=h1 : zr%=tque(i) : gosub 9900 :
     next i : return
2560 prompt$=format$(50) : in$="_" : ispa=7 : gosub
     3960
2565 for i=1 to anztit : ip(i+27)=1 : ip(i+40)=1 :
     nvar(2,tque(i))=h1 : zr%=tque(i) : gosub 9900 :
     next i : return
2570 prompt$=format$(51) : in$="__,__" : ispa=7 : gosub
     3960
2575 for i=1 to anztit : ip(i+27)=1 : ip(i+40)=1 :
     ivar(3,tque(i))=h1+1 : ivar(4,tque(i))=h2+1 :
     zr%=tque(i) : gosub 9900 : next i : return
2580 gosub 6800 : gosub 6500 : return
2590 gosub 4350 : return
rem
***********************arrays*************************
2600 if tcol<1 or tcol>7 or trow<1 or trow>5 then
     return
2610 on error goto 6000 : j=trow : i=trow : ileer=3
2620 j2=j+offo% : if j2>anztit then goto 2670
```

```
2625 ip(40+j2)=1 : ip(27+j)=1 : row%=10+j*2 :
     col%=int(icon(151+tcol*2)/100) : zr%=tque(j2)
2630 prompt$="?" : in$="__" : ispa=6 : gosub 3960 : if
     tcol<>1 then goto 2640
2632 tque(j2)=h1 : gosub 2300 : if tque(j2)>99 or
     tque(j2)<0 then goto 2620
2634 if tque(j2)>0 or j2>9 then goto 2640
2636 for j1=j2 to 9 : psym(j1)=psym(j1+1) :
     tque(j1)=tque(j1+1) : ip(40+j1)=1 : next j1 :
     anztit=anztit-1 : for j1=26 to 31 : ip(j1)=1 :
     next j1 : for j1=4 to 6 : ip(j1)=1 : next j1 :
     pointer%(11)=0
2640 if tcol=2 then psym(j2)=h1 : if psym(j2)>49 or
     psym(j2)<0 then goto 2620
2641 if tcol=3 then stift%(j2)=h1 : if stift%(j2)>8 or
     stift%(j2)<0 then goto 2620
2642 if tcol=4 then nvar(3,tque(j2))=h1 : gosub 9900
2643 if tcol=5 then nvar(1,tque(j2))=h1 : gosub 9900
2645 if tcol=6 then nvar(5,tque(j2))=h1 : gosub 9900
2647 if tcol=7 then t1=h1 : t2=h2 : if
     t2>ivar(5,tque(j2)) or t1>ivar(5,tque(j2)) or t1<0
     or t2<1 then return
2648 if tcol=7 then ivar(3,tque(j2))=t1+1 :
     ivar(4,tque(j2))=t2+1 : gosub 9900
2670 return
rem ***** routines performing softkey functions *******
2700 on a gosub 2720,2740,2760,2780,2800,2820,2840,2860
2710 ip(39)=1 : pointer%(12)=0 : return
2720 call alphaoff : call graph
2725 gosub 5000 : if a$=" " then 2725
2730 call graphoff : call alpha : return
2740 gosub 2900 : if taus=6 then forts=4 : pinit=1 :
     fort$=format$(60) : ifort=1 : running%=0 : return
```

```
2745 pinit=2 : forts=2 : fort$=plpgdr$+"pl " : ifort=1
     : ime=0 : running%=0 : return
2760 anzart=anzart+1 : if anzart>2 then anzart=1
2765 for i=26 to 31 : ip(i)=1 : next i : return
2780 ileer=45 : prompt$="Please enter new
     transformation" : col%=0 : row%=22 :
     in$=format$(25) : gosub 40000 : format$(25)=in$
     2790 in$=format$(26) : gosub 40000 :
     format$(26)=in$ : return
2800 prompt$="roll (-5/+5) " : in$="1" : ispa=2 : gosub
3960 : offo%=offo%+h1 : if offo%<0 then offo%=0
2805 if offo%>5 then offo%=5
2810 for i=27 to 32 : ip(i)=1 : next i : return
2820 gosub 2900 : pinit=1 : if taus=6 then forts=4 :
     fort$=format$(60) : ifort=1 : running%=0 : return
2825 forts=2 : fort$=plpgdr$+"pl" : ifort=1 :
     running%=0 : ime=0
2830 return
2840 gosub 7500 : for i=21 to 50 : ip(i)=1 : next i :
     return
2860 prompt$="Geben Sie die Register (von,bis) " :
     in$="1,50" : ispa=8 : gosub 3960 : if h1>100 or
     h2>100 or h1<0 or h2<0 or h1>h2 then goto 2860
2862 on error goto 6000 : print#11,format$(69);
     name$;"*****" : print#11, : print#11,format$(38)
2864 for zr%=h1 to h2 : gosub 9800 : gosub 5000 : gosub
     5000
2866 print#11, using format$(65);zr%,mid$(vat$,61,18),
     mid$(vat$,79,2),mid$(vat$,130,1),ivar(1,zr%),ivar(
     2,zr%),ivar(5,zr%),nvar(1,zr%),nvar(2,zr%),nvar(3,
     zr%),ivar(7,zr%),ivar(8,zr%)
2870 next zr% : return
rem  ******* procedure output to active device  *******
```

```
2900 for j=1 to anztit : gosub 2905 : next j : gosub
     6800
2902 for j=anztit+1 to 10 : dmax(j)=3 : next j : return
2905 if j<1 or ip(j+40)=0 or j>anztit then return
2906 if tmod=6 then gosub 19500 : return
2907 dmax(j)=2 : zr%=tque(j) : gosub 9200
2908 for j1=ivar(3,tque(j))-1 to ivar(4,tque(j))+1
2909 if j1<1 then dmax(j)=3 : dy(j,3)=0 : goto 2930
2910 if j1>ivar(5,tque(j))+1 then dy(j,dmax(j)+1)=0 :
     goto 2930
2911 Geschw=re(j1) : Substratkonz=re(j1+100)
2912 if tmod=1 then gosub 2940 : goto 2925
2913 if Substratkonz<=0 then goto 2930
2915 on tmod gosub 2940,2980,2960,2970,2950,2990
2922 goto 2925
2925 dmax(j)=dmax(j)+1: dx(j,dmax(j))=xwert :
     dy(j,dmax(j))=ywert
2927 if j1=ivar(4,tque(j))+1 then dmax(j)=dmax(j)-1
2930 next j1
2935 return
2940 xwert=Substratkonz : ywert=Geschw : return
2950 if Geschw=0 then : goto 2930
2955 xwert=Substratkonz : ywert=Substratkonz/Geschw
2957 return
2960 if Substratkonz=0 then : goto 2930
2965 xwert=Geschw : ywert=Geschw/Substratkonz
2967 return
2970 h2=Geschw/(nvar(3,tque(j))-Geschw): if h2<0 then
     goto 2930
2975 xwert=log(Substratkonz) : ywert=log(h2)
2977 return
2980 if Substratkonz=0 or Geschw=0 then goto 2930
2985 xwert=1/Substratkonz : ywert=1/Geschw
2987 return
```

```
2990 return
2995 rem ********** Data handling menu ***************
rem
3000 if anzeige>6 or anzeige<1 then anzeige=1
3002 on error goto 0 : if ime=3 and anzeige=pointer%
     (18) then 3020
3005 call cls : isk=99 : ime=3
3010 pointer%(18)=anzeige : def seg : get#14,4+anzeige
     : i1=varptr(#14) : call recscr(i1) : ip(59)=1 :
pointer%(13)=0 : ip(51)=1
3015 call offtouch : gosub 8500
3020 if isk=softk then return else h1$=format$(59+
     softk*2)+format$(60+softk*2) : gosub 8700 :
     isk=softk : return
3025 if pointer%(13)>30 then return
3030 pointer%(13)=pointer%(13)+1 : if pointer%(13)<18
     then gosub 3290 : return
3035 if pointer%(13)<20 then return
3040 ii=pointer%(13)-20 : if ip(ii+50)=0 then return
3042 on ii gosub
3950,3100,3100,3100,3100,3100,3100,3100,3395,6800
3045 ip(ii+50)=0 : return
3100 return
3290 zr%=pointer%(13)+(anzeige-1)*17 : if iu(zr%)=0
     then return
3295 gosub 3300 : return
rem  *************** directory line *****************
3300 if zr%>anzeige*17 or zr%<=(anzeige-1)*17 then goto
     3390
3305 col%=0 : row%=((zr%-1) mod 17)+5 : if row%>22 or
     row%<5 then goto 3390
3310 call curs(col%,row%) : if zr%>99 or zr%<1 then
     goto 3390
3315 gosub 9800
```

```
3320 if anzart=1 then print using format$(65);zr%,
     mid$(vat$,61,18),mid$(vat$,79,2),mid$(vat$,130,1),
     ivar(1,zr%),ivar(2,zr%),ivar(5,zr%),nvar(1,zr%),nv
     ar(2,zr%),nvar(3,zr%),ivar(7,zr%),ivar(8,zr%)
3330 iu(zr%)=0 : ip(60)=1
3390 return
3395 col%=0 : row%=22 : call curs(col%,row%) : print
     space$(79) : row%=0 : call curs(col%,row%) :
     return
rem *******routines for data handling menu commands ***
3400 col%=0 : row%=22 : call curs(col%,row%) : a=tsoft
3410 if tfeld>0 and tfeld<10 then gosub 3450
3420 if tsoft<1 or tsoft>8 then goto 3440
3430 if softk=1 then gosub 3700 else gosub 3500
3440 return
rem *************** single fields *****************
3445 if tfeld<4 and tfeld>0 then col%=int(icon
     (199+tfeld*2)/100) : row%=icon(199+tfeld*2)-
     col%*100+1 : call curs(col%,row%)
3450 prompt$="?" : ileer=8 : if tfeld=1 then
     in$=str$(xx) : gosub 40000 : xx=val(in$)
3455 if tfeld=2 then in$=str$(yy) : gosub 40000 :
     yy=val(in$)
3460 if tfeld=3 then in$=str$(zz) : gosub 40000 :
     zz=val(in$)
3495 gosub 3950 : return
rem * routines performing actions for second softkeys *
3500 ip(59)=1 : on tsoft gosub
3520,3540,3560,3580,3600,3620,3640,3660
3510 pointer%(13)=0 : return
3520 prompt$=format$(75) : in$="__,__" : ispa=5 : gosub
3960 : qra%=h1 : qrb%=h2
```

```
3522 call curs(col%,row%) : print space$(79) :
     prompt$=format$(77) : in$="__,__" : ispa=5 : gosub
     3960 : zrc%=h1 : anzop=h2
3524 call curs(col%,row%) : print space$(79) : if
     qra%<0 or qra%+anzop>100 or qrb%<0 or
     qrb%+anzop>100 or zrc%<0 or zrc%+anzop>100 then
     gosub 3395 : return
3526 For i=0 to anzop-1 : if nvar(5,qra%+i)<>
     nvar(5,qrb%+i) or nvar(5,qra%+i)=0 or
     nvar(5,qrb%+i)=0 then goto 3539
3530 gosub 8200
3539 next i : return
3540 gosub 6800 : for i=1 to 100 : iu(i)=1 : ip(i)=1 :
     next i : pointer%(20)=0
3545 fort$=inpgdr$+"iki" : running%=0 : ifort=1 :
     forts=3 : return
3560 prompt$=format$(78) : in$="__,_" : ispa=5 : gosub
3960 : zr%=h1 : smooth=h2
3565 if zr%>99 or zr%<1 or smooth<1 or smooth>3 then
     goto 3560
3570 call curs(col%,row%) : print "*** S M O O T H I N
     G ***";space$(50) : gosub 7000 : gosub 3395 :
     return
3580 prompt$="Geben Sie das Register" : in$="_" :
     ispa=3 : gosub 3960 : zr%=h1
3585 gosub 17000 : return
3600 for i=1 to 100 : ip(i)=1 : iu(i)=1 : next i :
     ime=0 : gosub 3000 : return
3620 gosub 16000 : return
3640 in$="reg?" : ispa=5 : gosub 3960 : zr%=h1 : gosub
     9800
3645 call curs(col%,row%) : print mid$(vat$,61,70) :
     for i=1 to 8 : print ivar(i,zr%);" "; : next i :
     print
```

```
3650 for i=1 to 8 : print nvar(i,zr%);"  "; : next i :
     print : ip(59)=0
3655 return
3660 softk=1 : gosub 3020 : return
rem * routines performing actions for first softkeys **
3700 ip(59)=1 : on tsoft gosub 3720,3740,3760,3780,
     3800,3820,3840,3860
3710 pointer%(13)=0 : return
3720 h1$=format$(9)+format$(10) : gosub 8700 : print
     "please select a softkey (Input2 =input for direct
     curve fit) "
3725 gosub 6800 : fort$=repgdr$+"rki" : running%=0 :
     ime=0 : ifort=1 : return
3740 if anzeige<=1 then anzeige=1 : return
3745 gosub 6800 : anzeige=anzeige-1 : gosub 3010 :
     return
3760 if anzeige>=6 then anzeige=6 : return
3765 gosub 6800 : anzeige=anzeige+1 : gosub 3010 :
     return
3780 prompt$=format$(71) : in$="__,__" : ispa=7 : gosub
     3960 : qr%=h1 : zr%=h2
3785 call curs(col%,row%) : print space$(79) : if
     qr%>99 or zr%>99 or qr%<1 or zr%<1 then goto 3780
3790 get #8,qr% : get #9,qr% : put #8,zr% : put #9,zr%
     : iu(zr%)=1 : gosub 9800 : return
3800 prompt$=format$(72) : in$="_" : ispa=3 : gosub
     3960 : zr%=h1
3805 if zr%>99  or zr%<1 then goto 3800
3810 get #8,zr% : call curs(col%,row%) :
     in$=mid$(vat$,61,18) : ileer=18 :
     prompt$=format$(73) : gosub 40000
3815 h1$=left$(vat$,60)+mid$(in$,1,18)+right$(vat$,52)
3817 lset vat$=h1$ : put #8,zr% : iu(zr%)=1 : return
```

```
3820 prompt$=format$(74) : in$="_" : ispa=3 : gosub
     3960 : zr%=h1 : if zr%<1 or zr%>99 then return
3830 get #8,zr% : hi$="*................ xx*.........
     *.................*.................K"
3835 for i=1 to 10 : ivar(i,zr%)=0 : nvar(i,zr%)=0 :
     next i
3838 gosub 9910 : return
3840 interpreter=1 : return
3860 softk=2 : gosub 3020 : return
rem *********** general subroutines *******************
3900 prompt$="?" : ileer=75 : in$=format$(jfo) : gosub
     40000 : format$(jfo)=in$ : return
3910 return
3920 close#1 : open "i",#1,filename$(1) : i=1
3925 while not eof(1)
3930 line input#1,h1$ : macro$(i)=h1$ : i=i+1 : if i>20
     then goto 3940
3935 wend
3940 return
3950 col%=0 : row%=2 : call curs(col%,row%) : h1$=" \
     \ #####.#### \                  \#####.#### \  \
     #####.####"
3955 print using h1$;"  C    =    A    *",xx,"    +
     B    *",yy," +",zz : return
rem ************ Input 1 odr two numbers **************
3960 ileer=ispa : gosub 40000
3970 a$="," : ikg=instr(in$,a$) : if ikg=0 then
     h1=val(in$) : h2=0 else h1=val(mid$(in$,1,ikg-1))
     : h2=val(mid$(in$,ikg+1,len(in$)-ikg))
3975 return
rem ******* return from external subroutine **********
4000 for i=1 to 10 : zr%=tque(i) : if dmax(i)<4 then
4080
4010  for j=3 to dmax(i) : re(j)=dx(i,j) : next j
```

```
4050  gosub 9100 : hi$="Ext."+str$(i)+space$(30) :
     gosub 9900
4080 next i
4090 return
rem *********** start MICHKIN software  ***************
4100 call cls : call graphoff : print "  MICHFIT
     software :   B.Michel   version   Feb. 02. 88"
4105 default=0  : anzeige=1 : autoscale=1 :
     interpolate=0 : anzart=1 : softk=1 : teart=1 :
     anzgem=1 : interpreter=0 : offo%=0
4110 comfile%=0 : pointer%(8)=1 : pointer%(20)=0 : for
     i=1 to 10 : interpk(i)=0 : stift%(i)=1 : next i :
     gosub 18700 : gosub 18500
4115 on error goto 4140 : open "i",#1,"a:"+name$+".KST"
     : for i=1 to 10 : input#1,tque(i) : next i : for
     i=1 to 10 : input#1,psym(i),stift%(i) : next i :
     print "reading ";name$+".KST"
4120 for i=1 to 40 : input#1,i1 : next i :
input#1,tein,teart, anzgem,anztit,tmod,taus,
     autoscale,interpolate,anzeige,anzart,tein,teart,an
     zgem,anztit,tmod,taus,interpolate,anzeige,anzart
4122 line input#1,format$(15) : line input#1,
     format$(16)
4125 pg(8)=0 : isav=anztit : anztit=0 : gosub 7525 :
     anztit=isav
4130 goto 4150
4140 resume 4145
4145 print name$+".KST not found" : on error goto 6000
4150 for i=1 to 90 : ip(i)=1 : next i : for i=91 to 99
     : ip(i)=0 : next i : pointer%(21)=0 :
     pointer%(23)=0
4155 if tein<3 then drive$=mid$(drt$,tein*2-1,2)
4160 if format$(52)<>"AUTO" then return
4170 macro$(1)=format$(53) : comfile%=1 : interpreter=1
```

```
4190 return
rem ********get directory**************************
4200 for i=8 to 11 : ip(i)=1 : next i : if
     pointer%(8)>5 or pointer%(8)<1 then pointer%(8)=1
4210 if tein>3 then inhalt$(1)=space$(240) : goto 4290
4220 for i=1 to 5 : inhalt$(i)="" : next i
4230 for i=1 to 30 : i1=i+(tein-1)*640+(teart-
     1)*128+(pointer%(8)-1)*24
4240 get#10,i1 : inhalt$(1)=inhalt$(1)+direntry$
4250 next i : inhalt$(1)=inhalt$(1)+space$(16)
4290 return
rem ***********background output*********************
4300 if pointer%(5)<1 then 4320
4310 if not eof(13) then input#13,io$ else gosub 4400 :
     return
4312 if len(io$)<10 then io$=io$+space$(32)
4315 print#12,io$
4320 return
4350 scol%=0 : srow%=22 : call curs(scol%,srow%) :
     print space$(79) : call curs(scol%,srow%) : print
     "No. of Plots :";pointer%(1);" repeat
     (0/1)";repeat%;" hit space " : a$=input$(1)
4355 call curs(scol%,srow%) : print space$(79) : call
     curs(scol%,srow%) : print "Please enter command
     for plot spool (e/d/r/c/p/s)" : a$=input$(1)
4360 if a$="d" then gosub 4400
4365 if a$="r" then repeat%=1
4370 if a$="c" then for j=1 to pointer%(1) : kill
     "PLOT"+chr$(j+64) : next j : pointer%(1)=0 :
     pointer%(5)=0
4375 if a$="p" then close#12 : close#13 : pointer%(5)=0
4380 if a$="s" then if pointer%(1)>0 then pointer%(5)=1
     : gosub 4415
4390 return
```

```
rem ***********close open Spool********************
4400 print#12,"PU PA 0,0;SP 0;" : close#12 : close#13 :
     if repeat%=1 then repeat%=0 : goto 4420
4402 kill "PLOT"+chr$(1+64) : pointer%(1)=pointer%(1)-1
     : if pointer%(1)=0 then pointer%(5)=0 : goto 4425
4405 for ips=1 to pointer%(1) : name "PLOT"+chr$(ips+
     1+64) as "PLOT"+chr$(ips+64) : next ips
4415 open "o",#12,"PLT" : width #12,128 : open
     "i",#13,"PLOT"+chr$(1+64)
4420 input#13,io$ : if io$="STOP" then scol%=0 :
     srow%=22 : call curs(scol%,srow%) : print "Hit
     Enter when Plotter ready (s=stop)" : a$=input$(1)
     : if a$="s" then pointer%(5)=0 : close#12 :
     close#13
4425 return
rem ********* Command interpreter ******************
4500 col%=0 : row%=27 : call curs(col%,row%)
4510 row%=22 : prompt$="" : ileer=79 : in$="" : gosub
     40000 : call curs(col%,row%) : print space$(159)
4520 if ia=189 or in$="e" or in$="E" then interpreter=0
     : return
4950 call curs(col%,row%) : print in$ : gosub 20005
4990 return
rem ************ Keyboard Touchscreen Mouse input  ****
5000 if touchs=0 and ilock=0 then print chr$(27);"-
     z2N"; : touchs=1 : rem **touchsense on ********
5005 iflag=0 : iplot=iplot+1 : if iplot>tshare then
     iplot=0 : if pointer%(1)>0 then gosub 4300
5010 a1$=inkey$ : if len(a1$)<1 then a$=" " : b=0 : a=0
     : return else a$=mid$(a1$,1,1) : a=asc(a$)-17 :
     b=asc(a$)-89 : iflag=1
5015 print chr$(27); "-z0N"; : touchs=0 : rem
     ******touchsense off **********************
```

```
5020 tfeld=0 : tarray=0 : tsoft=0 : tcol=0 : trow=0 :
     icol=0 : irow=0 : if eing%=1 then eing%=0 : goto
     5520
5025 if asc(a$)>17 and asc(a$)<27 then tsoft=asc(a$)-17
     : return
5027 if ilock=1 then return
5030 if a$="*" then gosub 6800 : gosub 1000 : return
5040 if a$="/" then gosub 6800 : gosub 2000 : return
5050 if a$="+" then gosub 6800 : gosub 3000 : return
5060 if a$="-" then gosub 6800 : pinit=1 : forts=2 :
     fort$=plpgdr$+"pl" : ifort=1 : running%=0 : return
5070 if a$="&" then interpreter=1 : in$="?__" : return
5080 if a$="?" then isme=ime : ihelp=ime+10 : gosub
6800 : gosub 6900 : return
5500 if a$="" then b$=inkey$ : print chr$(27);"a";
     chr$(17) : b$=input$(12) : icol=val(mid$(b$,5,2))
     : irow=val(mid$(b$,9,2))
5520 if ime<1 or ime>3 then return : rem
************sprung bei eing%=1*******
5525 for i=1 to 25 : i1=1+(i-1)*2+(ime-1)*100 : i2=i+65
5530  if icon(i1)=icon(i1+1) then 5600
5540  ocol%=icon(i1)/100 : orow%=icon(i1)-ocol%*100
5550  ucol%=icon(i1+1)/100 : urow%=icon(i1+1)-ucol%*100
5560  if icol<ocol% or icol>ucol% then 5580
5570  if irow>=orow% and irow<urow%+1 then tfeld=i :
     return
5580  if asc(a$)=i2 then tfeld=i : return
5590 next i
5600 ipoint=(ime-1)*100+51 : i2=89 : tarray=1
5610 i=ipoint : if icon(i)=0 or ipoint>(ime-1)*100+100
     then tarray=0 : return
5620 if icon(i)>9899 then rowoff%=icon(i)-9900 :
     anzz%=icon(i+1)/100 : anzs%=icon(i+1)-anzz%*100
     else tarray=0 : return
```

```
5630 for i=1 to anzs% : i1=ipoint+i*2 :
     ocol%=icon(i1)/100 : orow%=icon(i1)-ocol%*100
5640  ucol%=icon(i1+1)/100 : urow%=icon(i1+1)-ucol%*100
5650  for j=1 to anzz% : i2=i2+1 : orov%=orow%+(j-
     1)*rowoff% : urov%=urow%+(j-1)*rowoff%
5660  if icol<ocol% or icol>ucol% then 5680
5670  if irow>=orov% and irow<urov%+1 then tcol=i :
     trow=j : return
5680  if asc(a$)=i2 then tcol=i : trow=j : return
5690  next j
5700 next i : ipoint=ipoint+anzs%*2+2 : tarray=tarray+1
     : goto 5610
5990 return
rem ********Error Recovery ************************
6000 resume 6010
6010 close#1 : close#2 : close#3 : close#4 : close#5
6020 if err=0 then gosub 900 : rem garbage collection
6030 if erl>7500 and erl<8500 then print "please check
     if disk is (present/full/formatted)";err;erl; :
     goto 6200
6190 print "error "err;" ocurred on line ";erl;
6200 print " Hit any Key to proceed" : a$=input$(1)
6210 if a$="1" then gosub 1000 : goto 500
6220 if a$="2" then gosub 2000 : goto 500
6230 if a$="3" then gosub 3000 : goto 500
6240 if ime=1 then gosub 1000 : goto 500
6250 if ime=2 then gosub 2000 : goto 500
6260 if ime=3 then gosub 3000 : goto 500
6290 goto 500
rem *************warning ************************
6300 col%=0 : row%=22 : call curs(col%,row%) : print
     warn$;
6310 print "Hit any Key to proceed" : a$=input$(1)
6380 warnz%=0
```

```
6390 return
     rem ****clean string space if necessary***********
6400 i=fre(i1) : if i>ispace then return
6410 for i=1 to 10 : sym$(i)="" : next i : pointer%
     (25)=0
6420 for i=79 to 90 : format$(i)="" : next i :
     pointer%(26)=0
6490 i=fre("") : print i : return
rem ***************help menu**************************
rem
6500 page%=1 : upper%=1
6505 isme=ime : ime=4 : on error goto 6000 :
     h1$=format$(3)+format$(4) : gosub 8700
6510 close#1 : open "i",#1,"help.vsp" : call cls : call
     alphaoff : if page%=1 then 6550
6515 for j=1 to page%-1
6520   for i=1 to 44 : line input#1,h1$ : if eof(1)
     then 6590
6525   next i
6530 next j
6550 call cls : call alphaoff
6560 for i=1 to 44 : if eof(1) then 6580
6561  line input#1,h1$ : print h1$
6570 next i : if upper%=1 then row%=0 : call
     curs(col%,row%)
6580 call alpha
6590 return
rem ************ routines for softkey commands  *******
6700 if tsoft<1 or tsoft>8 then return
6705 on tsoft gosub
6710,6720,6730,6740,6750,6760,6770,6780
6707 return
6710 page%=page%-1 : gosub 6510 : return
6720 return
```

```
6730 page%=page%+1 : gosub 6550 : return
6740 page%=1 : gosub 6510 : return
6750 col%=0 : row%=45 : call curs(col%,row%) : Input
     "Please enter page (1-20)";page%
6755 call curs(col%,row%) : print space$(79) : gosub
6550 : return
6760 return
6770 if upper%=1 then upper%=0 : row%=46 else upper%=1
     : row%=0
6775 call curs(col%,row%) : return
6780 ime=0 : if isme>3 or isme<1 then isme=1
6782 close#1 : on isme gosub 1000,2000,3000 : x=fre("")
     : return
rem ****************store menu screen***************
6800 if ime>3 or ime<1 then return
6810 if ime=1 then istore=1 : ipp=20
6820 if ime=2 then istore=2 : ipp=40
6830 if ime=3 and anzeige>6 or ime=3 and anzeige<1 then
     anzeige=1 : return
6840 if ime=3 then istore=4+anzeige : ipp=60
6850 if ip(ipp)=0 then return
6890 def seg : i1=varptr(#14) : call stoscr(i1) :
     put#14,istore : return
rem ****************show help screen ******************
6900 call offtouch : call alpha : def seg :
     get#14,ihelp : i1=varptr(#14) : call recscr(i1)
6910 isme=ime : ime=5 : h1$=format$(5)+format$(6) :
     i1=5 : gosub 8700 : return
6950 if tsoft=1 then on isme gosub 1000,2000,3000 :
     return
6960 if tsoft=8 then ime=isme : gosub 6500
6990 return
7000 rem *******   smooth  ***************************
7005 for i=1 to 320 : re(i)=0 : next i
```

```
7010 gosub 9200 :  for j=64 to 1 step-1 : re(j+3)=re(j)
     : next j
7020 re(3)=2*re(4)-re(5) : re(2)=2*re(4)-re(6) :
     re(1)=2*re(4)-re(7) : j2=ivar(5,zr%)+1
7030 re(j2+4)=2*re(j2+3)-re(j2+2) : re(j2+5)=2*
     re(j2+3)-re(j2+1) : j1=1
7040 re(j2+6)=2*re(j2+3)-re(j2) : on smooth goto
7050,7100,7150
7050 for i=j1+3 to j2+3 : re(i-3)=(re(i-1)+re(i)+re(i
     +1))/3 : next i : goto 7240
7100 dd1=re(j1+2)-re(j1+1) : dd2=re(j1+3)-re(j1+2) :
     dd3=re(j1+4)-re(j1+3)
7105 de1=dd2-dd1 : de2=dd3-dd2 : df1=de2-de1
7110 for i=j1+3 to j2+3 : dd4=re(i+2)-re(i+1) :
     de3=dd4-dd3 : df2=de3-de2 : dg=df2-df1
7120 re(i-3)=re(i)-3*dg/35 : dd1=dd2 : dd2=dd3 :
     dd3=dd4 : de1=de2 : de2=de3 : df1=df2
7130 next i : goto 7240
7150 dd1=re(j1) : dd2=re(j1+1) : dd3=re(j1+2)
7155 for i=j1+3 to j2+3 : dd4=re(i) : re(i-3)=(-
     2*dd1+3*dd2+6*dd3+7*dd4+6*re(i+1)+3*re(i+2)-
     2*re(i+3))/21
7165 dd1=dd2 : dd2=dd3 : dd3=dd4 : next i
7240 for i=ivar(5,zr%)+2 to 64 : re(i)=0 : next i
7250 gosub 9100
7260 return
rem ************read plot file***********************
7500 col%=0 : row%=22 : prompt$="Please enter filename
     (max. 8 characters)" : in$="" : ileer=8 : gosub
     40000 : il=8 : gosub 7990 : if mid$(in$,1,4)
     ="EXIT" then return
7510 filename$(1)=drive$+in$+".P"+initialen$ : call
     curs(col%,row%) : print space$(79) : call
     curs(col%,row%)
```

```
7515 prompt$="Please enter first destination register"
     : in$="" : ileer=2 : gosub 40000 : tque(1)=val
     (in$) : if tque(1)<1 or tque(1)>99 then 7515
7520 call curs(col%,row%) : print space$(79) : j=1 : on
     error goto 6000 : close #1 : open "i",#1,filename
     $(j) : print "*** R E A D I N G *** file :
     ";filename$(j)
7522 ispace=1000 : gosub 6400 : rem check string space
7525 h1$=space$(250) : input#1,dform : if dform<>999
     then goto 7720
7530 for i=1 to 5 : input#1,pxu#(i),pyl#(i),pxo#
     (i),pyr#(i),psym(i) : next i
7535 for i=1 to 30 : input#1,pg(i),pgx(i),pgy(i),
     tl(i),pga(i) : next i
7540 for i=1 to 250 : input#1,w :  mid$(h1$,i,1)
     =string$(1,w) : next i
7542 for i=1 to 5 : mid$(beschr$(i),1,tl(i))=
     mid$(h1$,i*50-49,tl(i)) : next i
7545 for i=1 to 250 : input#1,w : mid$(h1$,i,1)
     =string$(1,w) : next i
7550 for i=1 to 250 : input#1,w : mid$(h1$,i,1)
     =string$(1,w) : next i
7552 for i=1 to 5 : mid$(beschr$(i+5),1,tl
     (i+5))=mid$(h1$,i*50-49,tl(i+5)) : next i
7555 input#1,tmod : taus=1 : idmax=0
7560 anztit=pg(8) : for i=6 to 10 : tl(i)=50 : next i
7570 for i=1 to anztit : tque(i)=tque(1)+i-1
7575 for j=1 to 10 : input#1,ivar(j,tque(i)) : next j
7580 for j=1 to 10 : input#1,nvar(j,tque(i)) : next j
7600 h1$=space$(70)
7610 for j=1 to 70 : input#1,w : mid$(h1$,j,1)=
     string$(1,w) : next j
7620 hi$=mid$(h1$,1,70) : zr%=tque(i)
7630 gosub 9910 : rem speichern einzelner Titrationsvar
```

```
7640 for j=1 to ivar(5,tque(i))+1 : input#1,re(j),
     re(j+100) : next j
7650 gosub 9100 : rem re() auf diskette schreiben
7660 next i
7690 close #1 : goto 7900
7720 if dform<>2265 then warnz%=7720 : warn$="Kein
     gueltiges Kinetikfile" : gosub 6300 : return
7740 for i=1 to 5 : input#1,pxu#(i),pyl#(i),pxo#(i),
     pyr#(i) : next i
7750 for i=1 to 30 : input#1,pg(i),pgx(i),pgy(i),tl(i),
     pga(i) : next i : input#1,idmax
7760 for i=1 to idmax : input#1,idx(i),idy(i),idp(i) :
     next i
7770 for i=1 to 20 : line input#1,h1$ : mid$(beschr
     $(i),1,tl(i))=mid$(h1$,1,tl(i)) : next i
7780 input#1,tmod,comfile% : taus=1 : anztit=pg(8) :
     autoscale=0 : default=1
7790 if tmod=6 then for i=1 to 2 : input#1,il : line
     input#1,h1$ : format$(i+24)=space$(il) :
     mid$(format$(i+24),1,il)=mid$(h1$,1,il) : next i
7792 if comfile%>0 then for i=1 to 20 : input#1,il :
     line input#1,h1$ : mid$(macro$(i),1,il)=h1$ : next
     i
7794 if comfile%>0 then for i=1 to 10 : input#1,il :
     line input#1,h1$ : mid$(p$(i),1,il)=h1$ : next i
7796 if comfile%>0 then for i=1 to 10 : input#1,c(i) :
     next i : if anztit=0 then 7900
7800 for i=1 to anztit : tque(i)=tque(1)+i-1 :
     zr%=tque(i) : in$=space$(130)
7810   for j=1 to 130 : input#1,w : mid$(in$,j,1)=
     string$(1,w) : next j
7815   lset vat$=in$ : put #8,zr% : gosub 9800 : gosub
     9950 : for j=1 to 164 : re(j)=0 : next j
7820   input#1,psym(i),stift%(i),stift%(i),interpk(i)
```

```
7830   for j=1 to ivar(5,tque(i))+1 : input#1,re(j),re
     (j+100) : next j
7840   gosub 9100 : rem re() auf diskette schreiben
7850 next i
7900 close#1 : for i=21 to 100 : ip(i)=1 : next i :
     ip(4)=1 : ip(5)=1 : ip(6)=1 : return
rem 7720 on error goto 2890 : close #1 : open
     "i",#1,drive$+a$+".P"+initialen$
rem 7730 for i=1 to 5 : input#1,h1,h2,h3,h4,psym(i) :
     pxu#(i)=h1 : pxo#(i)=h2 : pyr#(i)=h3 : pyl#(i)=h4
     : next i
rem 7740 for i=1 to 30 : input#1,pg(i),pgx(i),
     pgy(i),tl(i) : next i
rem 7741 for i=1 to 250 : input#1,w :  mid$(pbs$,
     i,1)=string$(1,w) : next i
rem 7742 for i=1 to 250 : input#1,w : mid$(h1$,i,1)
     =string$(1,w) : next i
rem 7743 input#1,tmod : taus=1 : for i=5 to 10 :
     tl(i)=50 : next i
rem 7744 gosub 19100 : anztit=pg(8)
rem 7750 for i=1 to anztit : tque(i)=tque(1)+i-1
rem 7751 input#1,ivar(1,tque(i)),ivar(2,tque(i)),
     ivar(3,tque(i)),ivar(4,tque(i)),ivar(5,tque(i))
rem 7752 input#1,nvar(1,tque(i)),nvar(2,tque(i)),
     nvar(3,tque(i)),nvar(4,tque(i)),nvar(5,tque(i))
rem 7753 input#1,ivar(6,tque(i)),ivar(7,tque(i)),
     ivar(8,tque(i)),nvar(9,tque(i)),nvar(10,tque(i))
rem 7800 n$=space$(70)
rem 7810 for j=1 to 70 : input#1,w : mid$(n$,j,1)
     =string$(1,w) : next j
rem 7820 hi$=mid$(n$,1,70) : zr%=tque(i)
rem 7830 gosub 9910 : rem speichern einzelner
     Titrationsvar
```

```
rem 7840 for j=1 to ivar(5,tque(i))+1 :
     input#1,re(j),re(j+100) : next j
rem 7850 gosub 9100 : rem re() auf diskette schreiben
rem 7860 next i
rem 7870 close #1 : drive$="a:" : goto 2000
rem *****************rescue overwritten file**********
7950 on error goto 7980 : h2$=plotfdr$+"RESCUE.P"
     +initialen$ : kill h2$
7960 on error goto 7985 : name plotfdr$+filename
     $(1)+".P"+initialen$ as h2$
7965 print "** R E S T O R E ** file ";filename$(1);"
     ** (old version renamed to RESCUE)"
7970 on error goto 6000 : return
7980 resume 7960
7985 resume 7970
rem ************UPPERCASE****************************
7990 for i=1 to il : ia=asc(mid$(in$,i,1)) : if ia>96
     and ia<123 then ia=ia-32 : mid$(in$,i,1)=chr$(ia)
7995 next i : return
rem ****************store plot file******************
8000 col%=0 : row%=21 : prompt$="Please enter filename
     (max. 8 characters)" : in$="" : ileer=8 : gosub
     40000 : il=8 : gosub 7990
8005 call curs(col%,row%) : print space$(79) : call
     curs(col%,row%) : if il<3 or il>8 then warnz%=8010
     : warn$="Invalid Filename" : return
8010 filename$(1)=in$+space$(8-il) : if mid$(filename
     $(1),1,4)="EXIT" or mid$(filename$(1),1,4)="PLOT"
     or mid$(filename$(1),1,6)="RESCUE" then print
     "illegal filename" : return
8015 for i=1 to inhp(idrive(11)*5) : il=(idrive(11)-
     1)*640+4*128+i : get#10,il
8017  if direntry$=mid$(filename$(1),1,8) then gosub
7950 : goto 8030
```

```
8020 next i : print "** S T O R I N G ** file
     ";filename$(1);" **"
8022 ispace=1000 : gosub 6400 : rem check string space
8025 il=(idrive(11)-1)*640+4*128+inhp(idrive(11)*5) :
     lset direntry$=mid$(filename$(1),1,8) : put#10,il
     : inhp(idrive(11)*5)=inhp(idrive(11)*5)+1 : gosub
     4200
8030 on error goto 6000 : close #1 : open
     "o",#1,plotfdr$+filename$(1)+"."+mid$(filekenn$,5,
     1)+initialen$ : pg(8)=anztit
8035 for j=1 to 20
8040  for i=1 to tl(j) : if mid$(beschr$(j),
     i,1)=chr$(13) then mid$(beschr$(j),i,1)=chr$(32)
8045  next i
8050 next j
8100 dform=2265 : print#1,dform
8105 for i=1 to 5 : print#1,pxu#(i),pyl#(i),
     pxo#(i),pyr#(i) : next i
8110 for i=1 to 30 : print#1,pg(i),pgx(i),pgy(i),
     tl(i),pga(i) : next i : print#1,idmax
8115 for i=1 to idmax : print#1,idx(i),idy(i),idp(i) :
     next i
8120 for i=1 to 20 : h1$=mid$(beschr$(i),1,tl(i)) :
     print#1,h1$ : next i
8125 print#1,tmod,comfile%
8130 if tmod=6 then for i=1 to 2 : il=len(format$
     (i+24)) : h1$=mid$(format$(i+24),1,il) :
     print#1,il : print#1,h1$ : next i
8140 if comfile%>0 then for i=1 to 20 :
     il=len(macro$(i)) : h1$=mid$(macro$(i),1,il) :
     print#1,il : print#1,h1$ : next i
8142 if comfile%>0 then for i=1 to 10 : il=len(p$(i)) :
     h1$=mid$(p$(i),1,il) : print#1,il : print#1,h1$ :
     next i
```

```
8144 if comfile%>0 then for i=1 to 10 : print#1,c(i) :
     next i : if anztit=0 then 8195
8150 for i=1 to anztit : zr%=tque(i) : gosub 9200 :
     gosub 9800 : h1$=vat$
8155 for j=1 to 130 : w=asc(mid$(h1$,j,1)) : print#1,w
     : next j
8160 print#1,psym(i),stift%(i),stift%(i),interpk(i)
8165 for j=1 to ivar(5,tque(i))+1 : print#1,re(j);re
     (j+100) : next j
8190 next i
8195 close #1 : taus=1 : ip(23)=1 : pointer%(12)=0 :
     return
rem ****************** Addition Multiplication********
8200 for j=1 to 10 : nvar(j,zrc%+i)=nvar(j,qra%+i) :
     next j
8205 for j=1 to 10 : ivar(j,zrc%+i)=ivar(j,qra%+i) :
     next j
8206  if ivar(5,qra%+i)<>ivar(5,qrb%+i) then x%=qrb% :
     y%=qra% : gosub 16010
8210 get #8,qra%+i : zr%=zrc%+i : put #8,zr% : gosub
     9800
8220 zr%=qra%+i : gosub 9200 : for j=1 to
     ivar(5,qra%+i)+1 : dy(1,j)=re(j) : next j
8230 if yy<>0 then zr%=qrb%+i : gosub 9200 : for j=1 to
     ivar(5,qra%+i)+1 : dy(2,j)=re(j) : next j
8240 for j=1 to ivar(5,qra%+i)+1
8250   if yy=0 then re(j)=dy(1,j)*xx+zz else
     re(j)=dy(1,j)*xx+dy(2,j)*yy+zz
8260 next j : zr%=zrc%+i : gosub 9100 :
     nvar(3,zr%)=nvar(3,qra%+i)*xx+nvar(3,qrb%+i)*yy+zz
     : gosub 9900 : gosub 3300
8290 return
rem **************** setup Touchscreen ***************
8500 rem
```

```
8540 for i=1 to 25 : i1=1+(i-1)*2+(ime-1)*100 : i2=i+65
8550  if icon(i1)=icon(i1+1) then 8600
8560  col%=icon(i1)/100 : row%=icon(i1)-col%*100
8570  colinc%=icon(i1+1)/100 : rowinc%=icon(i1+1)-
     colinc%*100-row% : colinc%=colinc%-col%
8580  resp$=chr$(i2) : call fntouch(row%,col%,
     rowinc%,colinc%,resp$)
8590 next i
8600 ipoint=(ime-1)*100+51 : i2=89
8610 i=ipoint : if icon(i)=0 then 8695
8620 if icon(i)>9899 then rowoff%=icon(i)-9900 :
     anzz%=icon(i+1)/100 : anzs%=icon(i+1)-anzz%*100
8630 for i=1 to anzs% : i1=ipoint+i*2 : col%=icon
     (i1)/100 : row%=icon(i1)-col%*100
8640  colinc%=icon(i1+1)/100 : rowinc%=icon(i1+1)-
     colinc%*100-row% : colinc%=colinc%-col%
8650  for j=1 to anzz% : i2=i2+1 : rov%=row%+(j-
     1)*rowoff%
8670   resp$=chr$(i2) : call fntouch(rov%,col%,
     rowinc%,colinc%,resp$)
8680  next j
8690 next i : ipoint=ipoint+anzs%*2+2 : goto 8610
8695 return
rem ****************Keylabel**************************
8700 if len(h1$)<160 then h1$=h1$+space$(161-len(h1$))
8705 for i=1 to 8 : m$=str$(i) : h2$=chr$(27)+"&f0a"
     +mid$(m$,2,1)+"k16d1L"+mid$(h1$,i*18-
     15,16)+chr$(17+i) : print h2$; : next i
8710 print chr$(27); "&jB"; : return
9100 rem ****** write re () to workfile  **************
9105 if zr%>100 or zr%<1 then zr%=1
9110 h4$=space$(253)
9120 for wl=1 to 63 : mid$(h4$,wl*4-3,4)=mks$(re(wl)) :
     next wl : lset e1$=h4$
```

```
9130 for wl=101 to 163 : mid$(h4$,(wl-100)*4-3,4)=mks$
     (re(wl)) : next wl
9135 lset e2$=h4$ : put #9,zr%
9140 return
9200 rem *********** read workfile to re () ***********
9205 if zr%>100 or zr%<1 then zr%=1
9210 get #9,zr%
9220 for wl=1 to 320 : re(wl)=0 : next wl
9230 for wl=1 to 63 : re(wl)=cvs(mid$(e1$,wl*4-3,4)) :
     next wl
9235 for wl=101 to 163 : re(wl)=cvs(mid$(e2$,(wl-
     100)*4-3,4)) : next wl
9240 return
9800 rem ********* read variables ********************
9802 if zr%>100 or zr%<1 then zr%=1
9805 get #8,zr%
9810 for wl=1 to 10
9820 ivar(wl,zr%)=cvi(mid$(vat$,wl*2-1,2)) :
     nvar(wl,zr%)=cvs(mid$(vat$,wl*4+17,4))
9830 next wl : return
9900 rem ******** write variables ********************
9902 if zr%>100 or zr%<1 then zr%=1
9905 get #8,zr% : hi$=mid$(vat$,61,70)+"           " :
9910 h4$=space$(130)
9915 for wl=1 to 10
9930 mid$(h4$,wl*2-1,2)=mki$(ivar(wl,zr%))
9940 mid$(h4$,wl*4+17,4)=mks$(nvar(wl,zr%))
9945 next wl : mid$(h4$,61,70)=mid$(hi$,1,70) : lset
     vat$=h4$ : put #8,zr%
rem ***************set update pointers*****************
9950 iu(zr%)=1 : pointer%(13)=0
9955 for wl=1 to 5 : if tque(wl+offo%)=zr% then
     ip(27+wl)=1 : pointer%(12)=0
9960 next wl
```

```
9965 for wl=1 to 10 : if tque(wl)=zr% then ip(wl+40)=1
     : pointer%(12)=0
9970 next wl : return
rem **************Operation Adjust ******************
16000 on error goto 6000 : col%=0 : row%=22 : call
     curs(col%,row%) : print "Adjust Register x to
     Register y"; : input x%,y%
16010 xmin=nvar(3,y%) : xmax=nvar(4,y%) :
     n=ivar(5,x%)+1
16015 zr%=x% : gosub 9200 : for i=1 to 500 :
     rex(i)=re(i) : next i
16020 zr%=y% : gosub 9200 : for i=1 to 500 :
     rey(i)=re(i) : next i : in=ivar(5,zr%)+1
16050 rex(in+1)=2*rex(in)-rex(in-1) :
     rex(in+101)=2*rex(in+100)-rex(in+99) :
     x0=2*rex(101)-rex(102) : y0=2*re(1)-re(2)
16090 i=0
16100 for j=1 to in : x=rey(j+100) : y=rey(j) : i=i+1
16110  for ii=1 to n+1 : if rex(ii)>x then iq=ii : goto
16150
16120  next ii : dy(1,i)=0 : goto 16165
16150  if iq=1 then dx=rex(1)-x0 : dy=re(1)-y0 else
     dx=rex(iq+100)-rex(iq+99) : dy=rex(iq)-rex(iq-1)
16160  if iq=1 then dy(1,i)=(x-x0)/dx*dy+y0 else
     dy(1,i)=(x-rex(iq-1))/dx*dy+rex(iq-1)
16165  dx(1,i)=rey(j+100)
16170 next j
16200 for i=1 to in : re(i)=dy(1,i) : re(i+100)=dx(1,i)
     : next i
16210 n=in
16260 for i=1 to 10 : nvar(i,zr%)=nvar(i,y%) :
     ivar(i,zr%)=ivar(i,y%) : next i
16300 gosub 9900 : gosub 9100
16400 return
```

```
rem *************data correction menu*****************
rem
17000 gosub 6800 : gosub 9200 : gosub 9800 : ime=6
17010 h1$=format$(7)+format$(8) : gosub 8700
17020 call cls : hi$=mid$(vat$,61,70) :
     iffs=int(ivar(5,zr%)/2)+1 : print format$(66)
17030 col%=0 : row%=1 : print : print : for i=1 to 8 :
     print ivar(i,zr%), : next i : print
17035 for i=1 to 8 : print nvar(i,zr%), : next i :
     print : print format$(68);format$(68)
17040 for i=1 to iffs : print using format$(67);i,
     re(i+100),re(i),i+iffs,re(i+100+iffs),re(i+iffs) :
     next i
17050 print space$(80) : return
17400 col%=0 : row%=2 : if tsoft<1 or tsoft>8 then
     return else call curs(col%,row%)
17410 on tsoft gosub
17520,17540,17560,17580,17600,17620,17640,17660
17450 return
17520 gosub 9910 : gosub 9100 : gosub 3000 : return
17540 prompt$="Please enter point " : in$="_" : ispa=4
     : gosub 3960 : i1=1 : i2=h1
17550 gosub 17700 : gosub 17020 : return
17560 prompt$="Please enter point " : in$="_" : ispa=4
     : gosub 3960 : i1=2 : i2=h1
17570 gosub 17700 : gosub 17020 : return
17580 prompt$="delete which point " : in$="_" : ispa=4
     : gosub 3960 : i2=h1
17585 for i=i2 to ivar(5,zr%) : re(i)=re(i+1) :
     re(i+100)=re(i+101) : next i :
     ivar(5,zr%)=ivar(5,zr%)-1
17590 gosub 17020 : return
17600 prompt$="insert which point " : in$="_" : ispa=4
     : gosub 3960 : i2=h1
```

```
17605 prompt$="Please enter x,y"+str$(i2) :
     in$="___,___" : ispa=12 : gosub 3960
17610 for i=ivar(5,zr%)+1 to i2+1 step-1 : re(i)=re(i-
     1) : re(i+100)=re(i+99) : next i :
     ivar(5,zr%)=ivar(5,zr%)+1 : re(i2+100)=h1 :
     re(i2)=h2
17615 gosub 17020 : return
17620 prompt$="?" : in$=hi$ : ileer=71 : gosub 40000 :
     hi$=in$ : if len(hi$)<70 then hi$=hi$+space$(70)
17630 gosub 9100 : gosub 17020 : return
17640 prompt$="Please enter integer variable" : in$="_"
     : ispa=2 : gosub 3960 : i2=h1 : if i2>8 or i2<1
     then return
17645 prompt$="Please enter variable no."+str$(i2) :
     in$=str$(ivar(i2,zr%)) : ispa=5 : gosub 3960 :
     ivar(i2,zr%)=h1 : return
17660 prompt$="Please enter normal variable" : in$="_"
     : ispa=2 : gosub 3960 : i2=h1 : if i2>8 or i2<1
     then return
17665 prompt$="Please enter variable no."+str$(i2) :
     in$=str$(nvar(i2,zr%)) : ispa=8 : gosub 3960 :
     nvar(i2,zr%)=h1 : return
17700 prompt$="Please enter new value no."+str$(i2) :
     if i1=1 then in$=str$(re(i2+100)) : ispa=8 : gosub
     3960 : re(i2+100)=h1
17710 if i1=2 then in$=str$(re(i2)) : ispa=8 : gosub
     3960 : re(i2)=h1
17720 return
rem ***********read default values ********************
18000 taus=1 : pxo#(5)=-1 : pyr#(5)=-1 : if tmod>6 then
     goto 18090
18010 i=1 : gosub 18510 : for i=27 to 30 : gosub 18510
     : next i
```

```
18020 for i=33 to 35 : gosub 18510 : next i : close#1 :
     pointer%(25)=0 : pointer%(26)=0
18060 tl(1)=len(format$(77+tmod*2)) : mid$(beschr$(1),
     1,tl(1))=format$(77+tmod*2) : tl(2)=len
     (format$(78+tmod*2)) : mid$(beschr$(2),
     1,tl(2))=format$(78+tmod*2) : default=1
18070 pxu#(1)=re(-9+tmod*10) : pxu#(2)=re(-8+tmod*10) :
     pxu#(3)=re(-7+tmod*10) : pxu#(4)=re(-6+tmod*10) :
     pxu#(5)=re(-5+tmod*10)
18080 pyl#(1)=re(-4+tmod*10) : pyl#(2)=re(-3+tmod*10) :
     pyl#(3)=re(-2+tmod*10) : pyl#(4)=re(-1+tmod*10) :
     pyl#(5)=re(tmod*10)
18090 return
18500 for i=1 to 37 : if i=33 or i=34 or i=35 or i=27
     or i=28 or i=29 or i=30 then 18505 else gosub
     18510
18505 next i : close#1 : return
18510 if i=1 then close#1 : open "r",#1,name$+".KSY",
     256  : field#1,256 as ex$
18520 get#1,i : if i>30 then 18560
18530  for ii=1 to 3 : l%=cvi(mid$(ex$,ii*84-83,2))
18540   format$(i*3+ii-3)=mid$(ex$,ii*84-81,l%)
18550  next ii : return
18560 get#1,i : if i>32 then 18600
18570  for ii=1 to 127 : icon((i-30)*127+ii-
     127)=cvi(mid$(ex$,ii*2-1,2))
18580  next ii : return
18600 get#1,i : if i>35 then return
18620  for ii=1 to 63 : re((i-32)*63+ii-
     63)=cvs(mid$(ex$,ii*4-3,4))
18640  next ii : return
18700 close#1 : open "r",#1,name$+".KSY",256  :
     field#1,256 as ex$
```

```
18710 get#1,36 : for ii=1 to 100 : if mid$(ex$,ii,1)=
     "0" then iu(ii)=0 else iu(ii)=1
18720  next ii : for ii=101 to 200 : if mid$(ex$,ii,1)
     ="0" then ip(ii-100)=0 else iu(ii-100)=1
18730  next ii : for ii=1 to 256 : h1$=h1$+"1" : next
     ii : lset ex$=h1$ : put#1,36 : close#1 : return
rem  **************************************************
rem                 mathematic interpreter
rem  **************************************************
rem  *************** interpreted transformation ********
19500 in$=format$(25) : il=len(in$) : zr%=tque(j)
19510 for i=1 to il : if mid$(in$,i,2)="RR" then
     h1$=str$(tque(j))+"  " : mid$(in$,i,2)
     =mid$(h1$,2,2)
19520 if mid$(in$,i,2)="P1" then h1$=str$(tque(j)+1)+"
     " : mid$(in$,i+1,2)=mid$(h1$,2,2)
19530 if mid$(in$,i,2)="M1" then h1$=str$(tque(j)-1)+"
     " : mid$(in$,i+1,2)=mid$(h1$,2,2)
19540 next i : rem print in$,
19550 gosub 20005
19600 in$=format$(26) : il=len(in$) : zr%=tque(j)
19610 for i=1 to il : if mid$(in$,i,2)="RR" then
     h1$=str$(tque(j))+"  " : mid$(in$,i,2)
     =mid$(h1$,2,2)
19620 if mid$(in$,i,2)="P1" then h1$=str$(tque(j)+1)+"
     " : mid$(in$,i+1,2)=mid$(h1$,2,2)
19630 if mid$(in$,i,2)="M1" then h1$=str$(tque(j)-1)+"
     " : mid$(in$,i+1,2)=mid$(h1$,2,2)
19640 next i : rem print in$
19650 gosub 20005
19900 return
rem  **************** get keyboard input *************
```

```
20000 col%=0 : row%=22 : ileer=75 : prompt$="" :
     in$=insav$ : gosub 40000 : if ia>16 and ia<27 or
     ia=189 then 25990
20005 il=len(in$) : gosub 7990 : zlev=0 : an%=0 : for
     i=1 to 10 : zr(i)=0 : next i :  intp$=in$ : if
     len(in$)<75 then insav$=in$ else insav$=mid$
     (in$,1,70)
20010 l=len(intp$) : if l<1 then goto 25990
20090 level=0 : lmax=0 : gpos=0 : on error goto 50000
rem  ************ analyse string  ******************
20100 for i=1 to l : h$=mid$(intp$,i,1)
20110 if h$="(" then level=level+1
20130 if level<0 then goto 29900
20140 if lmax<level then lmax=level
20150 if lmax=level and h$="(" then apos=i
20155 if lmax=level and h$=")" then zpos=i
20160 if h$=")" then level=level-1
20165 if h$="=" then gpos=i
20170 if asc(h$)<30 or asc(h$)>128 then goto 29900
20190 next i : rem print l,lmax,gpos
20200 if level<>0 or gpos=0 then goto 29900
20300 if lmax=0 then h1$=mid$(intp$,gpos+1,(l-gpos)) :
     apos=gpos : zpos=l+1 : goto 20500
20400 h1$=mid$(intp$,apos+1,(zpos-apos-1))
20500 opa=0 : for i=1 to len(h1$) : h$=mid$(h1$,i,1)
20510 if h$="^" then opo=i : opa=1 : goto 21000
20520 if h$="L" then opo=i : opa=2 : goto 21000
20530 next i
20550 for i=1 to len(h1$) : h$=mid$(h1$,i,1)
20560 if h$="*" then opo=i : opa=3 : goto 21000
20570 if h$="/" then opo=i : opa=4 : goto 21000
20580 if h$="D" then opo=i : opa=5 : goto 21000
20590 next i
20600 for i=1 to len(h1$) : h$=mid$(h1$,i,1)
```

```
20610 if h$="+" then opo=i : opa=7 : goto 21000
20620 if h$="-" then opo=i : opa=8 : goto 21000
20630 next i
20700 if lmax=0 then goto 25500
20710 if opa=0 then h2$=mid$(intp$,1,apos-1) :
    h3$=mid$(intp$,zpos+1,l-zpos-1) : intp$=h2$+h1$+
    h3$ : goto 20010
20720 goto 29800
21000 lpo=0 : rpo=0 : lrega=0 : if opo=1 then goto
    21050
21010 for i=opo-1 to 1 step-1 : h$=mid$(h1$,i,1)
21020 if h$="^" or h1$="L" or h$="*" or h$="/" or
    h$="+" or h$="-" or h$="D" then lpo=i : goto 21030
21025 next i : lpo=0
21030 for i=lpo+1 to opo-1 : h$=mid$(h1$,i,1)
21035 if lrega=0 then gosub 27000 : lque=ique :
    lrega=rega : lque2=nque2 : if rega<>0 then goto
    21050
21040 next i
21050 rrega=0
21055 for i=opo+1 to len(h1$) : h$=mid$(h1$,i,1) : if
    opo=len(h1$) then rpo=len(h1$)+1 : goto 21100
21060 if h$="^" or h1$="L" or h$="*" or h$="/" or
    h$="+" or h$="-" or h$="D" then rpo=i : goto 21070
21065 next i : rpo=len(h1$)+1
21070 for i=opo+1 to rpo-1 : h$=mid$(h1$,i,1)
21075 if rrega=0 then gosub 27000 : rque=ique :
    rrega=rega : rque2=nque2 : if rega<>0 then goto
21100
21080 next i
21100 h9$=space$(79) : mid$(h9$,lpo+1,1)=              m
    id$(str$(lrega),2,1) : mid$(h9$,opo+1,1)=mid$
    (str$(rrega),2,1)
```

```
21110 mid$(h9$,opo,1)=mid$(str$(opa),2,1) :
     mid$(h9$,opo+3,1)=mid$(str$(rque),2,1)
21120 mid$(h9$,lpo+3,1)=mid$(str$(lque),2,1) : rem
     print h9$
22000 gosub 26000 : if an%<0 then an%=0
22300 gosub 23000 : if an%<0 then an%=0
22500 if lpo=0 then h2$=" " else h2$=mid$(h1$,1,lpo)
22502 if rpo=len(h1$)+1 then h4$=" " else
     h4$=mid$(h1$,rpo,len(h1$)-rpo+1)
22504 h1b$=left$(intp$,apos) : h5$=mid$(intp$,zpos,l-
     zpos+1) : h3$="Z"+str$(zlev)
22510 if lpo=0 and rpo=len(h1$)+1 and lmax>0 then
     intp$=mid$(h1b$,1,len(h1b$)-1)+h3$+mid$(h5$,
     2,len(h5$)-1) : goto 22550
22515 if lpo=0 and rpo=len(h1$)+1 and lmax=0 then
     intp$=mid$(h1b$,1,len(h1b$))+h3$+mid$(h5$,2,len(h5
     $)-1) : goto 22550
22520 intp$=h1b$+h2$+h3$+h4$+h5$
22550 goto 20010 : rem print "Neuer String ";intp$ :
     goto 20010
rem  ************** perform operation on accumulator **
23000 for i=9  to 6 step-1 : if zr(i)=0 then zlev=i :
     zr(zlev)=an%+1 : goto 23050 : REM  N+1 !!!!
23010 next i
23050 on opa goto
23100,23200,23300,23400,23500,23600,23700,23800,23900
23080 for i=1 to an%+1 : dx(zlev,i)=rex(i) : next i :
     goto 24000
23100 for i=1 to an%+1 : if rex(i)<=0 then dx(zlev,i)=1
     else dx(zlev,i)=rex(i)^rey(i)
23105 next i : goto 24000
23200 for i=1 to an%+1 : if rex(i)<=0 or rey(i)<=0 then
     dx(zlev,i)=1 else dx(zlev,i)=log(rey
     (i))/log(rex(i))
```

```
23210 next i : goto 24000
23300 for i=1 to an%+1 : dx(zlev,i)=rex(i)*rey(i) :
     next i : goto 24000
23400 for i=1 to an%+1 : if rey(i)=0 then
     dx(zlev,i)=9.99E+20 else dx(zlev,i)=rex(i)/rey(i)
23420 next i : goto 24000
23500 dx(zlev,1)=((rex(1)+rex(2))/2-(3*rex(1)-
     rex(2))/2)/nvar(5,zr%)
23510 for i=2 to an% : h1=(rex(i-1)+rex(i))/2 :
     h2=(rex(i)+rex(i+1))/2
23520  dx(zlev,i)=(h2-h1)/nvar(5,zr%) : next i
23530 dx(zlev,an%+1)=(rex(an%+1)-(rex(an%+1)+rex
     (an%))/2)/nvar(5,zr%) : goto 24000
23600 rem
23700 for i=1 to an%+1 : dx(zlev,i)=rex(i)+rey(i) :
     next i : goto 24000
23800 for i=1 to an%+1 : dx(zlev,i)=rex(i)-rey(i) :
     next i : goto 24000
23900 rem
24000 rem print "zwischenresultat in register ";zlev;"
     : ";an%+1;" Punkte"  : for i=1 to an%+1 : print
     dx(zlev,i) : next i
24010 return
rem
25500 rem ******** ende op *********************
25505 h1$=intp$ : call curs(col%,row%) : rrega=0
25510 for i=gpos to len(h1$) : h$=mid$(h1$,i,1)
25520 if rrega=0 then gosub 27000  : rque=ique :
     rrega=rega : rque2=nque2
25530 next i : opo=gpos : rpo=len(h1$)+1
25540 lrega=7 : gosub 26000 : lrega=0
25550 for i=1 to gpos-1 : h$=mid$(h1$,i,1)
25560 if lrega=0 then gosub 27000 : lque=ique :
     lrega=rega : lque2=nque2
```

```
25570 if h$="D" then gosub 26500 : goto 25990
25575 if h$="P" then gosub 26600 : goto 25990
25580 next i
25620 if lrega<3 or lrega>10 then goto 25990 else print
     in$+space$(79-len(in$))
25630 on lrega goto
25800,25800,25650,25700,25750,25800,25850,25900,25960,
     25950
25650 print "X-Register";lque;" = ";an%;" Punkte" :
     zr%=lque : gosub 9200
25652 for i=1 to 10 : nvar(i,lque)=nvar(i,lzr%) :
     ivar(i,lque)=ivar(i,lzr%) : next i :
     hi$="Res."+str$(lzr%)+space$(30)
25654 ivar(5,zr%)=an% : gosub 9910
25656 for i=1 to an%+1 : re(i+100)=rey(i) : next i :
     gosub 9100
25660 goto 25990
25700 print "Y-Register";lque;" = ";an%;" Punkte" :
     zr%=lque
25702 for i=1 to 10 : nvar(i,lque)=nvar(i,lzr%) :
     ivar(i,lque)=ivar(i,lzr%) : next i : hi$="Res."
     +str$(lzr%)+space$(30)
25704 gosub 9910
25706 for i=1 to an%+1 : re(i)=rey(i) : next i : gosub
     9100
25710 goto 25990
25750 c(lque)=rey(1) : print "Konstante ";lque;" =
     ";rey(1) : goto 25990
25800 zr%=lque : if lque2<11 then ivar(lque2,zr%)=
     rey(1) : gosub 9900 : print "Variable
     ";lque;",";lque2;" = ";rey(1) : goto 25990
25810 zr%=lque : if lque2>10 then nvar(lque2-
     10,zr%)=rey(1) : gosub 9900 : print "Variable
     ";lque;",";lque2;" = ";rey(1) : goto 25990
```

```
25820 goto 25990
25850 if an%=0 then dx(j,3)=rey(1) : dmax(j)=3 : goto
25990 : print "1 Punkt"
25855 i1=2 : for i=ivar(3,lzr%)+1 to ivar(4,lzr%)+1 :
     i1=i1+1 : dx(j,i1)=rey(i-1) : next i : dmax(j)=i1
     : print dmax(j);" Punkte eingel"
25860 goto 25990
25900 if an%=0 then dy(j,3)=rey(1) : dmax(j)=3 : goto
25990 : print "1 Punkt"
25905 i1=2 : for i=ivar(3,lzr%)+1 to ivar(4,lzr%)+1 :
     i1=i1+1 : dy(j,i1)=rey(i-1) : next i : dmax(j)=i1
     : print dmax(j);" Punkte eingel"
25910 goto 25990
25950 zr%=lque : gosub 9200 : print "X-value";lque;",";
     lque2;" = ";rey(1),lque2 : if lque2<1 or lque2>64
     then 25990
25954 re(lque2+100)=rey(1) : gosub 9100 : goto 25990
25960 zr%=lque : gosub 9200 : print "Y-value";
     lque;",";lque2;" = ";rey(1),lque2 : if lque2<1 or
     lque2>64 then 25990
25964 re(lque2)=rey(1) : gosub 9100 : goto 25990
25990 return
rem ***********load accumulator ***********************
26000 on error goto 50000
26050 on lrega goto
26070,26080,26100,26110,26120,26130,26200,26200,26150,
     26160
26060 for i=1 to an%+1 : if opa=3 or opa=4 then
     rex(i)=1 else rex(i)=0
26062 if opa=2 then rex(i)=2.71828182
26065 next i : goto 26200
26070 hx$=mid$(h1$,lpo+1,opo-lpo-1)
26075 for i=1 to an%+1 : rex(i)=val(hx$) : next i :
     goto 26200
```

```
26080 if zr(lque)<an%+1 then an%=zr(lque)-1
26085 for i=1 to an%+1 : if zr(lque)=1 then i1=1 else
     i1=i
26090 rex(i)=dx(lque,i1)
26095 next i : zr(lque)=0 : goto 26200
26100 ziel=1 : zr%=lque : gosub 27200 : goto 26200 :
     rem get X
26110 ziel=1 : zr%=lque : gosub 27300 : goto 26200 :
     rem get Y
26120 for i=1 to an%+1 : rex(i)=c(lque) : next i : goto
26200
26130 if lque2<11 then for i=1 to an%+1 :
     rex(i)=ivar(lque2,lque) : next i : goto 26200
26140 if lque2>10 then for i=1 to an%+1 :
     rex(i)=nvar(lque2-10,lque) : next i : goto 26200
26150 zr%=lque : gosub 9200 : if lque2<1 or
     lque2>nvar(5,zr%) then rex(1)=0 : print "out of
     range" else rex(1)=re(lque2+100)
26155 for i=1 to an%+1 : rex(i)=rex(1) : next i : goto
     26200
26160 zr%=lque : gosub 9200 : if lque2<1 or
     lque2>nvar(5,zr%) then rex(1)=0 : print "out of
     range" : else rex(1)=re(lque2)
26165 for i=1 to an%+1 : rex(i)=rex(1) : next i : goto
26200
26200 nlinks%=an% : on rrega goto
26215,26220,26240,26250,26260,26270,26300,26300,26290,
     26295
26210 for i=1 to an%+1 : rey(i)=0 : next i : goto 26300
26215 hy$=mid$(h1$,opo+1,rpo-opo-1) : for i=1 to an%+1
     : rey(i)=val(hy$) : next i : goto 26300
26220 if zr(rque)<an%+1 then an%=zr(rque)-1
26225 for i=1 to an%+1 : if zr(rque)=1 then i1=1 else
     i1=i
```

```
26230 rey(i)=dx(rque,il)
26235 next i : zr(rque)=0 : goto 26300
26240 ziel=2 : zr%=rque : gosub 27200 : goto 26300
26250 ziel=2 : zr%=rque : gosub 27300 : goto 26300
26260 for i=1 to an%+1 : rey(i)=c(rque) : next i : goto
26300
26270 if rque2<11 then for i=1 to an%+1 : rey(i)=ivar
     (rque2,rque) : next i : goto 26300
26280 if rque2>10 then for i=1 to an%+1 : rey(i)=
     nvar(rque2-10,rque) : next i : goto 26300
26290 zr%=rque : gosub 9200 : if lque2<1 or
      lque2>nvar(5,zr%) then rey(1)=0 : print "out of
      range" else rey(1)=re(lque2+100)
26292 for i=1 to an%+1 : rey(i)=rey(1) : next i : goto
      26300
26295 zr%=lque : gosub 9200 : if lque2<1 or
      lque2>nvar(5,zr%) then rey(1)=0 : print "out of
      range" : else rey(1)=re(lque2)
26297 for i=1 to an%+1 : rey(i)=rey(1) : next i : goto
      26300
26300 nrechts%=an% : rem print "akku a = ";rex(1);"
      akku b = ";rey(1)
26310 if nlinks%=nrechts% then goto 26360
26320 if nlinks%=0 then for i=1 to nrechts%+1 :
      rex(i)=rex(1) : next i : nlinks%=nrechts% :
      an%=nlinks%
26330 if nrechts%=0 then for i=1 to nlinks%+1 :
      rey(i)=rey(1) : next i : nrechts%=nlinks% :
      an%=nrechts%
26340 if nlinks%<nrechts% then an%=nrechts%
26350 if nrechts%<nlinks% then an%=nlinks%
26360 return
rem ***********display print **********************
```

```
26500 print in$+space$(79-len(in$)) : if an%=0 then
     print rey(1) : goto 26590
26501 h1$=space$(79) : for i=1 to 8 : mid$(h1$,i*9,2)
     =str$(i)+" " : next i : print h1$ : h1$=space$(79)
     26502 for i=1 to an% : i1=int(i/8) : i2=i-i1*8 :
     if i2=0 then i2=8
26505 h2$=str$(rey(i))+"          " : a$="E" :
     ie=instr(h2$,a$) : if ie>4 then h3$=h2$ :
     mid$(h2$,5,6)=mid$(h3$,ie,6)
26507 mid$(h1$,i2*9-3,8)=h2$ : if i2=8 then
     mid$(h1$,1,4)=str$(i-7)+"  " : print h1$ :
     h1$=space$(79)
26510 next i : print h1$
26590 return
26600 print#11,in$ : if an%=0 then print#11,rey(1) :
     goto 26690
26601 h1$=space$(79) : for i=1 to 8 : mid$(h1$,i*9,2)
     =str$(i)+" " : next i : print#11,h1$ :
     h1$=space$(79)
26602 for i=1 to an% : i1=int(i/8) : i2=i-i1*8 : if
     i2=0 then i2=8
26605 h2$=str$(rey(i))+"          " : a$="E" :
     ie=instr(h2$,a$) : if ie>4 then h3$=h2$ :
     mid$(h2$,5,6)=mid$(h3$,ie,6)
26607 mid$(h1$,i2*9-3,8)=h2$ : if i2=8 then
     mid$(h1$,1,4)=str$(i-7)+"  " : print#11,h1$ :
     h1$=space$(79)
26610 next i : print#11,h1$
26690 return
27000 on error goto 50000 : rem *************
     registerart **************
27010 rega=0 : ique=0 : nque2=0
```

```
27011 if h$="1" or h$="2" or h$="3" or h$="4" or h$="5"
     or h$="6" or h$="7" or h$="8" or h$="9" or h$="0"
     then rega=1
27012 if h$="Z" then rega=2 : ique=val(mid$(h1$,i+1,2))
27014 if h$="X" then rega=3 : ique=val(mid$(h1$,i+1,2))
27016 if h$="Y" then rega=4 : ique=val(mid$(h1$,i+1,2))
27018 if h$="C" then rega=5 : ique=val(mid$(h1$,i+1,2))
27020 if h$="V" then rega=6 : ique=val(mid$(h1$,i+1,2))
     : nque2=val(mid$(h1$,i+4,2))
27028 if h$="A" then rega=7 : ique=val(mid$(h1$,i+1,2))
27030 if h$="B" then rega=8 : ique=val(mid$(h1$,i+1,2))
27040 if h$="N" then rega=9 : ique=val(mid$(h1$,i+1,2))
     : nque2=val(mid$(h1$,i+4,2))
27050 if h$="M" then rega=10 : ique=val(mid$
     (h1$,i+1,2)) : nque2=val(mid$(h1$,i+4,2))
27090 return
27200 an%=ivar(5,zr%) : lzr%=zr% : gosub 9200 : rem
     ****************** get xreg******************
27235 for i=1 to ivar(5,zr%)+1
27250 if ziel=1 then rex(i)=re(i+100)
27260 if ziel=2 then rey(i)=re(i+100)
27270 next i : return
27300 an%=ivar(5,zr%) : lzr%=zr% : gosub 9200 : rem
     ********** get yreg******************
27340 for i=1 to ivar(5,zr%)+1
27350  if ziel=1 then rex(i)=re(i)
27360  if ziel=2 then rey(i)=re(i)
27370 next i : return
29800 print "Interpreter error" : goto 25990
29900 print "wrong input" : goto 25990

rem ****************input routine*********************
```

```
40000 icol%=col%+len(prompt$) : il=len(in$) : if
     il>ileer or icol%>79 then in$=space$(ileer) :
     return
40020 if icol%+ileer>80 then ileer=80-icol%
40030 if il<ileer then in$=in$+space$(ileer-il)
40040 call curs(col%,row%) : print prompt$;in$;"*" :
     call curs(icol%,row%) : ia=0 : tshare=100
40050 while ia<>13 : call echo(ia) : if ia>17 and ia<27
     or ia=189 then 40190
40060 iplot=iplot+1 : if iplot>tshare then iplot=0 : if
     pointer%(1)>0 then gosub 4300
40080 wend
40100 h1$=space$(80)+"" : call readli(row%,h1$) :
     in$=mid$(h1$,icol%+1,ileer) : h1$=""
40190 return
rem ***************Error Recovery ***************
50000 resume 50010
50010 col%=0 : row%=22 : call curs(col%,row%)
50020 print "Interpretererror  "err;" ocurred on line
     ";erl
50200 print "Hit any Key to proceed" : a$=input$(1)
50270 goto 20000

rem  **************************************************
rem  *******   MICHFIT software :   iki.bas        ****
rem  *******   Author : Bruno Michel       *************
rem  **************************************************
rem
rem
rem
rem
rem
rem
rem
```

```
rem
rem
*******************************************************
rem                    Start of source code
rem
********************************************************
rem  *************** definitions ********************
90 rem $include : 'comvar'
220 width 255 :
222 dim p$(10)
223 dim interpk(10),isvar(8),nsvar(8)
225 dim rex(500),rey(500),re(400),var(5,20),
     hs$(5),tnc(5),dio$(5),sr%(5,5),swe(5,5),swr(5,5),f
     ak%(5,5)
227 dim lbeg(5),lend(5),lstep(5),p(5)
230 xx=1: yy=1: zz=0 : smooth=2    : var(1,1)=0
370 field#8,130 as vat$ : field#9,256 as e1$,256 as e2$
410 close #11 : open "LPT1:" for output as #11 len=80
rem *********** start depending on forts ************
450 rem
490 if forts=3 then gosub 3000 : goto 550
500 gosub 1500 : forts=2
550 if warn%=1 then forts=1
560 close#1,#2,#3,#4,#5
565 for i=1 to 5 : dio$(i)="" : hs$(i)="" : next i
570 for i=1 to 20 : filename$(i)="" : next i
575 wout$=""
580 sva$="" : n$="" : hi$="" : in$="" : ex1$="" :
     ex2$=""
585 rem
590 col%=0 : row%=22 : call curs(col%,row%) : print
     fre("")
600 chain pret$
1500 rem
```

```
1600 if tein=4 then gosub 1900 : forts=2 : return
1760 on teart goto 1764,1768,1772,1776,1778
1764 call cls
1766 for j=1 to anztit : zr%=tque(j) : gosub 15000 :
     next j : return
1768 call cls
1770 for j=1 to anztit : zr%=tque(j) : gosub 13000 :
     next j : return
1772 gosub 10000 : return
1776 gosub 10000 : return : rem  einlesen und Mitteln
1778 return
rem col%=0 : row%=22 : call curs(col%,row%) : print
     space$(79) : j=1 : gosub 7520 : gosub 2000 :
     return
rem 1780 print format$(12) : gosub 5000 : if a$="E" or
     a$="e" then return
rem 1782 on error goto 6000 : gosub 1660 : if
     touchf>inhp(teart) or touchf<1 then return
rem 1785 h1$=drive$+mid$(inhalt$(teart),touchf*8-
     7,8)+"."+mid$(filekenn$,teart,1)+initialen$ :
     h2$=drive$+"rescue.sys" : kill h2$ : name h1$ as
     h2$ : print#11,"File ";h1$;" geloescht"
rem 1790 for i=touchf to inhp(teart) : mid$(inhalt$
     (teart),touchf*8-7,8)=mid$(inhalt$(teart),
     touchf*8+1,8) : next i : inhp(teart)=inhp(teart)-1
rem 1795 gosub 1200 : return
rem ********* Keyboard input routine *****************
1900 zr%=tque(1) : in$="Name der Daten  " : ispa=18 :
     gosub 1990 : hi$=mid$(in$,1,18)
1910 ispa=2 : in$="* Bezeichnung zB. a1 " : gosub 1990
     : hi$=hi$+mid$(in$,1,2) : ispa=12 : in$="Datum  "
     : gosub 1990 : hi$=hi$+mid$(in$,1,10)
1915 ispa=20 : in$="Enzym Name " : gosub 1990 :
     hi$=hi$+mid$(in$,1,20)
```

```
1917 ispa=10 : in$=in$+" Konzentration :" : gosub 3960
     : nvar(1,zr%)=h1
1918 ispa=20 : in$="Substrat Name " : gosub 1990 :
     mid$(in$,20,1)="K" : hi$=hi$+mid$(in$,1,20)
1919 ispa=10 : in$=in$+" Konzentration :" : gosub 3960
     : nvar(2,zr%)=h1
1920 in$="Messbereich (2 Eingaben)  " : gosub 3960 :
     ivar(1,zr%)=h1 : ivar(2,zr%)=h2 : in$="Normierung
     1=on 0=off" : gosub 3960 : ivar(6,zr%)=h1
1922 if ivar(6,zr%)=1 then in$="Normierbereich  " :
     gosub 3960 : ivar(7,zr%)=h1 : ivar(8,zr%)=h2 else
     ivar(7,zr%)=0 : ivar(8,zr%)=0
1925 in$="Anzahl Messpunkte  " : gosub 3960 :
     ivar(5,zr%)=h1 : ivar(3,zr%)=1 : ivar(4,zr%)=
     ivar(5,zr%)+1 : km=10 : vmax=100
1930 for i=1 to ivar(5,zr%)+1 : col%=0 : in$="X-Wert
     und Y-Wert"+str$(i-1)+" " : gosub 3960 :
     re(i+100)=h1 : re(i)=h2 : row%=row%+1 : if row%>42
     then row%=22
1940 next i : col%=0 : row%=22
1950 nvar(3,zr%)=0 : i1=0 : nvar(3,zr%)=re(ivar
     (5,zr%)+1) : gosub 9100 : gosub 9910 : return
1990 call curs(col%,row%) : print in$;space$(ispa) :
     col%=col%+len(in$) : in$="?"+space$(ispa-1)
1995 gosub 40000 : col%=0 : if mid$(in$,1,4)="exit" or
     mid$(in$,1,4)="EXIT" then print "abgebrochen " :
     goto 500
1997 return
rem *************swap workfile*******************
3000 on error goto 6000 : col%=0 : row%=22 : call
     curs(col%,row%) : print space$(79) : ileer=2
3010 col%=12 : call curs(col%,row%) : print "Enter name
     of workfile to be read"
```

```
3020 col%=0 : in$="b:________" : gosub 40000 : if
     len(in$)<>10 or mid$(in$,2,1)<>":" then 3000
3030 filename$(1)=in$
3100 col%=0 : row%=22 : call curs(col%,row%) : print
     space$(79)
3110 col%=12 : call curs(col%,row%) : print "Enter name
     of workfile to be written"
3120 col%=0 : in$="b:________" : gosub 40000 : if
     len(in$)<>10 or mid$(in$,2,1)<>":" then 3100
3130 filename$(2)=in$
3150 if filename$(1)=filename$(2) then print "file
     cannot be copied to itself" : goto 3000
3160 rem
3170 col%=0 : row%=22 : call curs(col%,row%) : print
     space$(79)
3200 if mid$(filename$(2),3,8)=space$(8) then 3500
3205 call curs(col%,row%) : print "
     ";filename$(2)
3210 rem
3220 close #2 : open "r",#2,filename$(2)+".KVA",130 :
     field#2,130 as wout$
3230 for i=1 to 99 : call curs(col%,row%) : print
     "store";i;"   "
3240   get#8,i : h1$=mid$(vat$,130,1) : if h1$="K" or
     h1$=" " or h1$="B" or h1$="." or h1$="R" then 3250
     else 3800
3250   lset wout$=vat$ : put#2,i
3260 next i
3310 rem
3320 close #2 : open "r",#2,filename$(2)+".KIW",512 :
     field#2,256 as ex1$,256 as ex2$
3330 for i=1 to 99 : call curs(col%,row%) : print
     "store";i;"   "
3340   get#9,i
```

```
3350   lset ex1$=e1$ : lset ex2$=e2$ : put#2,i
3360 next i
3390 close#2
3500 if mid$(filename$(1),3,8)=space$(8) then 3720
3505 call curs(col%,row%) : print "
     ";filename$(1)
3510 close #1 : open "r",#1,filename$(1)+".KVA",130 :
     field#1,130 as wout$
3520 rem
3530 for i=1 to 99 : call curs(col%,row%) : print "read
     ";i;"   "
3540   get#1,i : h1$=mid$(wout$,130,1) : if h1$="K" or
     h1$=" " or h1$="B" or h1$="." or h1$="R" then 3550
     else 3850
3550   lset vat$=wout$ : put#8,i
3560 next i : i1=99
3610 close #1 : open "r",#1,filename$(1)+".KIW",512 :
     field#1,256 as ex1$,256 as ex2$
3620 rem
3630 for i=1 to i1 : call curs(col%,row%) : print "read
     ";i;"   "
3640   get#1,i
3650   lset e1$=ex1$ : lset e2$=ex2$ : put#9,i
3660 next i
3690 rem
3700 close#1 : close#2
3710 for i=1 to 99 : gosub 9800 : next i
3720 call curs(col%,row%) : print space$(79)
3750 forts=3
3790 return
3800 print "active workfile is invalid no file stored :
     operation cancelled"
3810 kill filename$(2) : close#1 : close#2 : goto 3750
```

```
3850 print "invalid workfile on disk : read stopped at
     file";i-1 : i1=i-1 : if i>1 then 3610
3860 close#1 : close#2 : goto 3750
rem **************** Input 1 or 2 number  *************
3960 call curs(col%,row%) : print in$;space$(ispa) :
     col%=col%+len(in$) : in$="?"+space$(ispa-1) :
     gosub 40000 : col%=0
3970 a$="," : ikg=instr(in$,a$) : if ikg=0 then
     h1=val(in$) : h2=0 else h1=val(mid$(in$,1,ikg-1))
     : h2=val(mid$(in$,ikg+1,len(in$)-ikg))
3975 return
rem **********background output**********************
4300 if pointer%(5)<1 or iplot<0 then 4320
4305 if eof(13) and aufspulen%=1 then 4320
4310 if not eof(13) then input#13,io$ else gosub 4400 :
     return
4315 print#12,io$
4320 return
rem ***********close open Spool************************
4400 print#12,"PU PA 0,0;SP 0;" : close#12 : close#13 :
     kill "PLOT"+str$(1) : pointer%(1)=pointer%(1)-1 :
     if pointer%(1)=0 then pointer%(5)=0 : goto 4425
4405 for ips=1 to pointer%(1) : name "PLOT"+str$(ips+1)
     as "PLOT"+str$(ips) : next ips
4415 open "o",#12,"PLT" : width #12,128 : open
     "i",#13,"PLOT"+str$(1)
4420 input#13,io$ : if io$="STOP" then scol%=0 :
     srow%=22 : call curs(scol%,srow%) : print "Hit
     Enter when Plotter ready (s=stop)" : a$=input$(1)
     : if a$="s" then pointer%(5)=0 : close#12 :
     close#13
4425 return
rem ************* Command interpreter *****************
rem 4500 col%=0 : row%=27 : call curs(col%,row%)
rem 4510 row%=22 : gosub 40000 : call curs(col%,row%) :
     print space$(159)
rem 4520 if ia=189 or in$="e" or in$="E" then
     interpreter=0 : return
rem 4950 call curs(col%,row%) : print in$ : gosub 20005
rem 4990 return
rem **********Keyboard Touchscreen Mouse input  *******
5000 print chr$(27);"-z2N"; : rem
     **********************touchsense on ********
5005 iplot=iplot+tshare : if iplot>tshare then iplot=-
     99
5007 iflag=0 : if pointer%(1)>0 then gosub 4300
5010 a1$=inkey$ : print chr$(27); "-z0N"; : rem
     ******touchsense off ************
5015 if len(a1$)<1 then a$=" " : b=0 : a=0 : goto 5990
     else a$=mid$(a1$,1,1) : a=asc(a$)-17 : b=asc(a$)-
     89 : iflag=1
5020 if ilock=1 then return
5022 tfeld=0 : tarray=0 : tsoft=0 : tcol=0 : trow=0 :
     icol=0 : irow=0
5025 if asc(a$)>17 and asc(a$)<27 then tsoft=asc(a$)-17
     : return
rem 5030 if a$="*" then gosub 1000 : return
rem 5040 if a$="/" then gosub 2000 : return
rem 5050 if a$="+" then gosub 3000 : return
5060 if a$="-" then pret$="vsp" : pinit=1 : forts=2 :
     fort$="pl" : ifort=1 : running%=0 : return
5070 if a$="&" then interpreter=1 : in$="?__" : return
5500 if a$="" then b$=inkey$ : print
     chr$(27);"a";chr$(17) : b$=input$(12) :
     icol=val(mid$(b$,5,2)) : irow=val(mid$(b$,9,2))
5520 for i=1 to 25 : i1=1+(i-1)*2+(ime-1)*100 : i2=i+65
5530  if icon(i1)=icon(i1+1) then 5600
5540  ocol%=icon(i1)/100 : orow%=icon(i1)-ocol%*100
```

```
5550  ucol%=icon(i1+1)/100 : urow%=icon(i1+1)-ucol%*100
5560  if icol<ocol% or icol>ucol% then 5580
5570  if irow>=orow% and irow<urow%+1 then tfeld=i :
     return
5580  if asc(a$)=i2 then tfeld=i : return
5590 next i
5600 ipoint=(ime-1)*100+51 : i2=89 : tarray=1
5610 i=ipoint : if icon(i)=0 or ipoint>(ime-1)*100+100
     then tarray=0 : return
5620 if icon(i)>9899 then rowoff%=icon(i)-9900 :
     anzz%=icon(i+1)/100 : anzs%=icon(i+1)-anzz%*100
     else tarray=0 : return
5630 for i=1 to anzs% : i1=ipoint+i*2 :
     ocol%=icon(i1)/100 : orow%=icon(i1)-ocol%*100
5640  ucol%=icon(i1+1)/100 : urow%=icon(i1+1)-ucol%*100
5650  for j=1 to anzz% : i2=i2+1 : orov%=orow%+(j-
     1)*rowoff% : urov%=urow%+(j-1)*rowoff%
5660  if icol<ocol% or icol>ucol% then 5680
5670  if irow>=orov% and irow<urov%+1 then tcol=i :
     trow=j : return
5680  if asc(a$)=i2 then tcol=i : trow=j : return
5690  next j
5700 next i : ipoint=ipoint+anzs%*2+2 : tarray=tarray+1
     : goto 5610
5990 return
rem  ****************** error recovery  ***************
6000 resume 6010
6010 print "Ein Fehler "err;" ist auf Zeile ";erl;"
     aufgetreten ";
6200 print "Hit any Key to proceed" : a$=input$(1) :
     forts=1
6290 goto 560
9100 rem ******** write re () to workfile  ************
9105 if zr%>100 or zr%<1 then zr%=1
```

```
9110 n$=space$(253)
9120 for wl=1 to 63 : mid$(n$,wl*4-3,4)=mks$(re(wl)) :
     next wl : lset e1$=n$
9130 for wl=101 to 163 : mid$(n$,(wl-100)*4-
     3,4)=mks$(re(wl)) : next wl
9135 lset e2$=n$ : put #9,zr%
9140 return
9200 rem *********** read re () from workfile  ********
9205 if zr%>100 or zr%<1 then zr%=1
9210 get #9,zr%
9220 for wl=1 to 320 : re(wl)=0 : next wl
9230 for wl=1 to 63 : re(wl)=cvs(mid$(e1$,wl*4-3,4)) :
     next wl
9235 for wl=101 to 163 : re(wl)=cvs(mid$(e2$,(wl-
     100)*4-3,4)) : next wl
9240 return
9800 rem ****** read variables from workfile **********
9802 if zr%>100 or zr%<1 then zr%=1
9805 get #8,zr%
9810 for wl=1 to 10
9820 ivar(wl,zr%)=cvi(mid$(vat$,wl*2-1,2)) :
     nvar(wl,zr%)=cvs(mid$(vat$,wl*4+17,4))
9830 next wl : return
9900 rem ******** write variables to workfile *********
9902 if zr%>100 or zr%<1 then zr%=1
9905 get #8,zr% : hi$=mid$(vat$,61,70)+"          " :
9910 n$=space$(130)
9915 for wl=1 to 10
9930 mid$(n$,wl*2-1,2)=mki$(ivar(wl,zr%)) :
     mid$(n$,wl*4+17,4)=mks$(nvar(wl,zr%))
9940 next wl : mid$(n$,61,70)=mid$(hi$,1,70)
9950 lset vat$=n$ : put #8,zr% : iu(zr%)=1
9960 return
```

```
rem *********** read in and average *******************
10000 call cls
10010 for j1=1 to anzgem : gosub 11000 : next j1 : rem
     read Variables
10030 gosub 11300 : rem display und check
10040 if iende=1 then 10350
10050 for j1=1 to 2 : gosub 11600 : next j1
10070  gosub 12000 : rem read 3 Bereiche Seriell und
     store parallel
10250 for j2=1 to 2
10260  zr%=tque(j2) : gosub 9200
10290  i1=0 : nvar(3,tque(j2))=0
10300  for j=ivar(5,tque(1))-1 to ivar(5,tque(1))+1
10310   i1=i1+1 : nvar(3,tque(j2))=nvar
     (3,tque(j2))+re(j)
10320  next j : nvar(3,tque(j2))=nvar(3,tque(j2))/i1 :
     print "Maximalwert der Titrationskurve ";j2;" =
     ";nvar(3,tque(j2))
10330  gosub 9900 : next j2
10350 return
rem *************read Variables from input file *******
11000 close#j1 : on error goto 6000 : open "i",#j1,
     filename$(j1)
11010 line input#j1,hs$(j1) : if len(hs$(j1))<137 then
11010 else line input#j1,h1$
11040 for i=1 to 20 : var(j1,i)=val(mid$(h1$,i*12-
     11,11)) : next i
11042 line input#j1,h1$
11044 line input#j1,h1$
11060 ivar(5,tque(1))=var(1,5) : nvar(4,tque(1))=
     var(1,6) : nvar(1,tque(1))=var(1,3)
11065 enzk$=mid$(hs$(1),43,20) : enzs$=mid$
     (hs$(1),63,20)
11070 return
```

```
rem ***********Display und check variables ************
11300 col%=0 : row%=1 : call curs(col%,row%)
11305 for j1=1 to anzgem : print hs$(j1)
11310   for i1=1 to 6
11315     for i=1 to 10 : print using
      "####.##";var(j1,(i1-1)*10+i); : next i : print
11317   next i1
11320 next j1 : print "k = Korrektur f= fortsetzen
      e=exit" : iende=0
11325 a$=input$(1)
11330 if a$="a" then iende=1 : goto 11500
11420 if a$="f" then goto 11500
11430 if a$="k" then goto 11460
11460 input "Geben Sie die Kinetik (1-5) und die
      Variable";i,i1 : row%=0
11480 print var(i,i1) : input var(i,i1) : goto 11325
11500 return
rem ** write input variables to internal variables ****
11600 zr%=tque(j1) : rem ************ store
      ***********************************
11610 nvar(1,zr%)=var(1,3) : nvar(2,zr%)=var(1,14)
11620 ivar(5,zr%)=int(var(1,5)) : nvar(4,zr%)=var(1,14)
      : ivar(3,zr%)=1 : ivar(4,zr%)=int(ivar(5,zr%)+1)
11630 ivar(6,zr%)=1 : ivar(7,zr%)=var(1,10) :
      ivar(8,zr%)=var(1,11) : ivar(4,zr%)=int(ivar
      (5,zr%)+1)
11640 nvar(3,zr%)=0 : nvar(5,zr%)=1 : nvar(9,zr%)=0 :
      nvar(10,zr%)=0 : nvar(6,zr%)=0 : ivar(10,zr%)=0
11650 if j1=1 then ivar(1,zr%)=var(1,1) :
      ivar(2,zr%)=var(1,2) else ivar(1,zr%)=var(1,12) :
      ivar(2,zr%)=var(1,13)
11690 hi$=filename$(1)+space$(18-len(filename$(1)))+
      mid$(hs$(1),1,1)+mid$(hs$(1),3,1)+mid$(hs$(1),5,10
      )+mid$(hs$(1),43,39)+"K"
```

```
11700 gosub 9910
11750 return
12000 rem *** read and average kinetic files***********
12010 for j2=1 to 320 : re(j2)=0 : next j2
12020 for j1=1 to anzgem
12070  for j2=1 to ivar(5,tque(1))+1 : line
     input#j1,dio$(j1)
12080   in$=mid$(dio$(j1),1,9)+mid$(dio$(j1),11,2) :
     re(j2)=re(j2)+val(in$) : print "*";in$;"*";"
     ";re(j2),
12082   in$=mid$(dio$(j1),13,9)+mid$(dio$(j1),23,2) :
     re(j2+200)=re(j2+200)+val(in$) : print
     "*";in$;"*";" ";re(j2+200),
12084   in$=mid$(dio$(j1),25,9)+mid$(dio$(j1),35,2) :
     re(j2+100)=re(j2+100)+val(in$) : print
     "*";in$;"*";" ";re(j2+100)
12090  next j2 : close #j1 : dio$(j1)=" " : hs$(j1)=" "
12100 next j1
12110 for j2=1 to 320 : re(j2)=re(j2)/anzgem : next j2
12130 zr%=tque(1) : gosub 9100 : rem Store Kinetik 1
12140 for j2=1 to 64 : re(j2)=re(j2+200) : next j2
12150 zr%=tque(2) : gosub 9100  : rem store kinetik 2
12200 return
rem ****************************************************
13000 return
rem ************read X-Y data file ********************
15000 on error goto 6000 : if j=1 then 15020
15010 if filename$(j)=filename$(j-1) then 15050
15020 close#1 : open "i",#1,filename$(j) : inmod=1
15050 eflag=0 : input#1,hs$(1) : if len(hs$(1))<3 then
     hs$(1)=hs$(1)+"    "
15052 if mid$(hs$(1),1,3)="***" then print hs$(1) :
     inmod=2 : n=500 : goto 15060
```

```
15053 if mid$(hs$(1),1,3)="///" then print hs$(1) :
     inmod=1 : goto 15050
15054 n=val(mid$(hs$(1),1,3)) : print hs$(1)
15055 if n>500 or n<1 then print "n ist
     falsch";n;"*";hs$(1);"* a=abort r=retry" :
     a$=input$(1) : if a$="r" then 15050 else 15400
15060 for i=1 to n : input#1,hs$(1) : if
     mid$(hs$(1),1,3)="///" or mid$(hs$(1),1,3)="***"
     then 15100
15062  if inmod=2 then 15075
15065  hs$(2)=mid$(hs$(1),1,9)+mid$(hs$(1),11,2) :
     hs$(3)=mid$(hs$(1),14,9)+mid$(hs$(1),24,2)
15070  rex(i)=val(hs$(2)) : rey(i)=val(hs$(3)) : print
     rex(i),rey(i) : goto 15090
15075  i2=len(hs$(1)) : for i1=1 to i2 : if
     mid$(hs$(1),i1,1)="=" then 15085
15080  next i1 : goto 15090
15085  rex(i)=val(mid$(hs$(1),1,i1-1)) :
     rey(i)=val(mid$(hs$(1),i1+1,i2-i1))
15090 if eof(1) then i=i+1 : goto 15100
15095 next i
15100 if inmod=2 then n=i-1
15105 if n<63 then goto 15200
15110 print "Anzahl Punkte is zu gross (>63)";n
15120 print "Bitte geben Sie Anfangspunkt, Endpunkt,
     und Schrittweite (0,0,0=exit)"
15125 print "Bei 200 Punkten zB. 1,120,2"
15130 input t1,t2,t3 : if t3=0 then goto 15400
15135 if (t2-t1)/t3>63 or t2<t1+1 or t1<1 or t2>n then
15110
15140 i1=0
15150 for i=t1 to t2 step t3 : i1=i1+1
15160  if t3>2 then rex(i1)=(rex(i-1)+rex(i)+rex
     (i+1))/3 else rex(i1)=rex(i)
```

```
15165  if t3>2 then rey(i1)=(rey(i-1)+rey(i)+rey
     (i+1))/3 else rey(i1)=rey(i)
15170 next i : n=i1
15200 for i=1 to 64 : re(i)=0 : re(i+100)=0 : next i
15210 for i=1 to n
15220  re(i+100)=rex(i) : re(i)=rey(i)
15230 next i
15280  for j2=1 to 10 : nvar(j2,zr%)=0 : ivar(j2,zr%)=0
     : next j2
15290  gosub 9100 : rem store Kinetik parallel
15295  ivar(5,zr%)=n : ivar(4,zr%)=n+1 : ivar(3,zr%)=1
     : nvar(2,zr%)=re(n+100) : nvar(4,zr%)=re(n+100)
15300  filename$(j)=filename$(j)+space$(20) :
     hi$=mid$(filename$(j),1,18)+space$(49)+"K"
15310  gosub 9910 : rem speichern einzelner spektren
     Variablen
15400 return
40000 rem ************keyboard input routine***********
40010 ip=1 : ins=0 : on error goto 40600 : con%=73 :
     rox%=row%
40020 call curs(col%,row%) : print in$;space$(ileer) :
     call curs(col%,row%) : ilmax=len(in$)
40100 com%=col%+ip-1 : call curs(com%,row%) : ilock=1 :
     gosub 5000 : ilock=0 : if len(a1$)=1 then goto
     40120
40105 if len(a1$)=2 then goto 40120
40110 goto 40100
40120 if a1$=chr$(27) then ins=1 : a1$=" " else ins=0
40130 if a1$=chr$(127) then goto 40500
40200 ia=asc(a1$)
40210 if ia=13 or ia=189 then goto 40300
40220 if ia=9 then if ip<ilmax+1 then ip=ip+1 :
     com%=col%+ip-1 : goto 40100 else goto 40100
```

```
40230 if ia=8 then if ip>1 then ip=ip-1 : com%=col%+ip-
     1 : goto 40100 else goto 40100
40250 if ip<=ilmax and ins=0 then mid$(in$,ip,1)=al$ :
     print al$ : ip=ip+1 : goto 40100
40260 if ip<=ilmax and ins=1 then in$=in$+" " : for
     ii=len(in$)-1 to ip step-1 : mid$(in$,ii+1,1)=
     mid$(in$,ii,1) : next ii : mid$(in$,ip,1)=al$ :
     goto 40020
40270 in$=in$+al$ : ilmax=ip : ip=ip+1 : print al$ :
     goto 40100
40300 return
40500 if ip>len(in$) then goto 40100
40510 for ii=ip to len(in$)-1 : mid$(in$,ii,1)=
     mid$(in$,ii+1,1) : next ii
40520 hl$=in$ : l=len(in$)-1 : in$=mid$(hl$,1,l) : goto
     40020
40600 resume 40610
40610 print "Eingabefehler  "err;" ist auf Zeile
     ";erl;" aufgetreten ";
40620 print "Hit any Key to proceed" : a$=input$(1)
40630 goto 40000

rem    ************************************************
rem    *****   MICHFIT software :  rki.bas    **********
rem    *****   Author : Bruno Michel         ***********
rem
*******************************************************
rem
rem
rem
rem
rem
rem
rem
```

```
rem
rem
rem
rem
rem
*******************************************************
rem                    Start of source code
rem
********************************************************
rem  ****************** definitions  ******************
90 rem $include : 'comvar'
220 width 255
225 dim re(400),ireg(11,11),erww(11,11),
     raster(11,11),fak%(11,11)
226 dim rex(330),rey(330),zr(10),ik(10),ist(10),
     interpk(10)
227 dim lbeg(11),lend(11),lstep(11),fit%(11),
     pne%(11),da%(11)
232 drucker%=1 : b$="e" : offset%=0 : anzan%=0
250 xx=1: yy=1: zz=0 : smooth=2 : running%=1 : iplot=0
     : ilock=0 : pointer%(26)=0
260 exp$="+#.###^^^^"
270 idmax=pointer%(2) : comfile%=pointer%(3) :
     interpreter=pointer%(4) : dbez$="a:b:c:"
rem ******************Kinetic Strings **************
300 format$(79)="*** direct curve fit / simulation ***
     10 = user :"
332 format$(80)=" 1 = Michaelis Menten    4 = Hill
     equation        7 = Inhibition (ki)  10=own"
334 format$(81)=" 2 = Consecutive react.  5 =
     Noncooperative sites 8 = Unncomp. inh.   Formula"
336 format$(82)=" 3 = 2 km 2 Vmax values  6 =
     Sequential interact. 9 = Noncomp. inh. "
```

```
338 format$(83)="11 = Dependen site m.   14 = Exchange
     mechanism  17 = Lineweaver Burke"
340 format$(84)="12 = Independent site m 15 = Var.
     inhibitor      18 = Eadie Hofstee "
342 format$(85)="13 = Dead end complex   16 =       19
     = Hannes Woolfe "
344 remod$="Michaelis Menten   Consecutive react.  2
     km 2 Vmax values  Hill equation
     Noncooperative sitesSequential interact.Inhibition
     (ki)     Uncomp. inh.        Noncomp. inh.        "
346 remoe$="Dependen site mechan. Independent site
     MDead end complex    Exchange mechanism  Var.
     Inhibitor      Linear Regression   Lineweaver
     Burke    Eadie Hofstee       Hannes Woolf         "
350 filename$(1)="VmaxPa1 VmaxVmaxVmaxVmaxVmaxVmax
     VmaxPa1 k1o k1o k1o k1o VmaxVmaxVmaxVmaxVmax"
351 filename$(2)="Km  Pa2 Km1 K   Km1 Km1 Ki  Km1 Km1
     Pa2 K1  K1  K1  K1  Ki     *   *   *  "
352 filename$(3)="    Pa3 Vm2 Nh  *   *   VinhKi  Ki
     Pa3 k2o k2o k2o ke  Vres   *   *   *  "
353 filename$(4)="    Pa4 Km2     *   *       *   *
     Pa4 K2  K2  K2  k3                    "
354 filename$(5)="               Pa5                "
355 filename$(6)="               Pa6                "
356 geb$="Please enter : "
360 pres$="\                  \  :  ####.## nM      \
     \ Max. :  #####.## uM"
365 pres2$="data points used from : ###  to   ###
     register with absolute deviations ##  "
370 pres3$="regression type       :  "
400 if pinit<>1 then goto 900
410 field#8,130 as vat$ : field#9,256 as e1$,256 as e2$
420 close #11 : open "LPT1:" for output as #11 len=80
rem  **************************************************
rem                         Main program
rem  ***************************************************
500 while running%
505 rem if warnz%>0 then gosub 6300
510 rem if comfile%>0 then a$=macro$(comfile%) :
     comfile%=comfile%+1 : if comfile%>21 then
     comfile%=0
515 rem if comfile%>0 then gosub 4500
520 rem if comfile%=0 and interpreter=1 then gosub 4500
550 tshare=100 : ileer=1 : ilock=0 : if comfile%=0 and
     interpreter<>1 then gosub 11000 : rem
     *******warte auf Eingabe **
560 if ime<1 or ime>3 then ime=1
565 if iflag=0 then 790
570 gosub 3400
790 wend
rem
*******************************************************
rem                Subroutines / procedures
rem
******************************************************
rem **************clean string space*****************
800 h$="" : h1$="" : h2$="" : h3$="" : h4$="" : h5$=""
     : h6$="" : h9$="" : h1b$="" : hx$="" : hy$=""
805 geb$="" : zest$="" : fsp$="" : hj$="" : hk$="" :
     hl$="" : hm$="" : sz$=""  : pres$="" : pres2$="" :
     pres3$=""
810 in$="" : intp$="" : form$="" : warn$="" : remod$=""
     : remoe$=""
815 for i=78 to 90 : format$(i)="" : next i : for i=1
     to 20 : filename$(i)="" : next i
820 i=fre("")
825 for i=41 to 50 : ip(i)=1 : next i : pointer%(13)=0
```

```
830 pointer%(2)=idmax : pointer%(3)=comfile% :
    pointer%(4)=interpreter : chain pret$
900 end
rem ****************Directory line ******************
3300 if zr%>anzeige*17 or zr%<=(anzeige-1)*17 then goto
3390
3305 col%=0 : row%=((zr%-1) mod 17)+5 : if row%>22 or
     row%<5 then goto 3390
3308 call curs(col%,row%) : print space$(79)
3310 call curs(col%,row%) : if zr%>99 or zr%<1 then
     goto 3390
3320 iu(zr%)=1 : print using format$(65);zr%,mid$(vat
     $,61,18),mid$(vat$,79,2),mid$(vat$,130,1),ivar(1,z
     r%),ivar(2,zr%),ivar(5,zr%),nvar(1,zr%),nvar(2,zr%
     ),nvar(3,zr%),ivar(7,zr%),ivar(8,zr%)
3390 return
3395 col%=0 : row%=22 : call curs(col%,row%) : print
     space$(159) : col%=0 : row%=0 : call
     curs(col%,row%)
3397 return
rem *routines executing commands from regression menu *
3400 rem
3410 row%=22 : call curs(col%,row%) : if tsoft>0 and
     tsoft<9 then 3500
3490 return
rem ******* routines performing softkey functions *****
3500 on tsoft gosub
3520,3540,3560,3580,3600,3620,3640,3660
3510 return
3520 gosub 6000
3530 gosub 3395 : return
3540 row%=22 : call curs(col%,row%) : print space$(159)
     : call curs(col%,row%)
3542 input "which position in batch job (1-10)";ji
```

```
3545 gosub 4900 : row%=26 : call curs(col%,row%) :
     print space$(79) : gosub 5000
3550 gosub 3395 : return
3560 for j=1 to 5
3562 for i=1 to 10 : ireg(j,i)=0 : erww(j,i)=0 :
     raster(j,i)=0 : fak%(j,i)=0 : next i
3564 next j
3566 for ji=1 to 10 : gosub 4900 : row%=26 : call
     curs(col%,row%) : print space$(79)
3568 gosub 5000
3570 next ji : gosub 3395 : return
3580 gosub 5200
3590 gosub 3395 : return
3600 gosub 10000 : return
3610 return
3620 offset%=offset%+3 : if offset%>3 then offset%=0
3630 return
3640 col%=0 : row%=22 : print space$(239) : print
     space$(159) : call curs(col%,row%)
3645 print "Please enter new Formula eg.
     R=P1/(1+XRR/P2)" : row%=23 : prompt$="" : ileer=79
3647 in$=format$(26) : gosub 40000 : gosub 7990 :
     format$(26)=in$
3650 gosub 3395 : return
3660 forts=2 : pinit=1 : running%=0 : return
rem ************error***************************
4200 resume 4210
4210 col%=0 : row%=22 : call curs(col%,row%) : print
     "error No. ";err;" occurred on line ";errl;
4220 a$=input$(1) : goto 3400
4250 resume 4260
4260 on error goto 4270
4265 print#11, "regression ";ji;" terminated with error
     ";err;" on line ";errl
```

```
4270 on error goto 4250 : goto 5400
rem ****************************************************
4900 row%=4 : col%=0 : call curs(col%,row%)
4902 row%=22 : call curs(col%,row%) : print space$(239)
     : print space$(159) : call curs(col%,row%)
4905 row%=27 : call curs(col%,row%) : row%=22 : rem
     *************************
4910 call curs(col%,row%) : print format$(79); : if
     len(format$(26))>30 then print
     mid$(format$(26),1,29);".." else print format$(26)
4920 print format$(80+offset%) : print
     format$(81+offset%) : print format$(82+offset%) :
     return
rem *************direct curve fit *********************
5000 call curs(col%,row%) : input "Enter type of
     regression 0=exit, register and points
     (from/to)";ireg(1,ji),ireg(2,ji),ireg(3,ji),ireg(4
     ,ji) : zr%=ireg(2,ji) : : if ireg(1,ji)<1 or
     ireg(1,ji)>19 then gosub 3395 : goto 500
5070 if zr%>99 or zr%<1 or ireg(3,ji)<1 or ireg(4,ji)<2
     or ireg(3,ji)>ivar(5,zr%)-1 or ireg(4,ji)>ivar
     (5,zr%) then goto 5000
5100 regart=ireg(1,ji)
5102 row%=22 : call curs(col%,row%) : print space$(239)
     : print space$(159) : call curs(col%,row%)
5105 call curs(col%,row%) : print "register";zr%;"
     regression type";ireg(1,ji);"from/to";
     ireg(3,ji);ireg(4,ji)
5107 row%=23 : call curs(col%,row%) : input "Please
     enter destination register and max. number of
     cycles";ireg(6,ji),ireg(7,ji) : if ireg(6,ji)<0 or
     ireg(6,ji)>99 then goto 5107
5110 row%=24 : if ireg(1,ji)>16 then ireg(5,ji)=0 :
     goto 5190
```

```
5120 call curs(col%,row%) : input "Please enter
     register containing absolute deviation (0=equally
     weighted)";ireg(5,ji) : if ireg(5,ji)>99 or
     ireg(5,ji)<0 then goto 5110
5130 if ireg(5,ji)<>0 and ivar(5,zr%)<>ivar(5,ireg
     (5,ji)) then call curs(col%,row%) : print "invalid
     register  " : w=1 : call wart(w) : goto 5120
5135 row%=22 : call curs(col%,row%) : print space$(239)
     : print space$(159) : call curs(col%,row%)
5150 for j=1 to 6 : row%=21+j : h1$=mid$(filename$(j),
     regart*4-3,4) : if h1$="    " then erww(j,ji)=1 :
     raster(j,ji)=1  : fak%(j,ji)=-1 : goto 5165
5155 prompt$=h1$+":  new estimate, max. accuracy and
     factor :" : in$=str$(nvar(4+j,zr%)) : gosub 7100 :
     if mid$(in$,1,1)="e" then 5165
5156 erww(j,ji)=h1 : raster(j,ji)=h2 : fak%(j,ji)=h3 :
     i1=h3 : if i1>-1 and i1<5000 then iparm=j
5160 if i1=-1 or i1=0 or i1=1 or i1=2 or i1=4 or i1=8
     or i1=16 or i1=32 or i1=64 or i1=128 or i1=256 or
     i1=512 or i1=1024 or i1=2048 or i1=4096 then goto
5165 else goto 5155
5165 next j : ivar(10,zr%)=ireg(1,ji)
5170 row%=22 : call curs(col%,row%) : print space$(239)
     : print space$(159) : call curs(col%,row%)
5190 return
5200 flag%=0 : i1=0 : row%=22 : call curs(col%,row%)
5210 input "Please enter position in batch job to start
     regression (1-10)";ii : if ii>10 or ii<1 then goto
     5210
5220 for ji=ii to 10 : regart=ireg(1,ji) : stdevreg=
     ireg(5,ji) : zr%=ireg(2,ji) : von%=ireg(3,ji) :
     bis%=ireg(4,ji)
5225 if regart<1 or regart>19 then goto 5450
```

```
5232 stdevreg=ireg(5,ji) : if stdevreg=0 then for i=1
     to 64 : dy(1,i)=1 : next i : goto 5238
5235 zr%=stdevreg : gosub 9200 : for i=1 to 64 :
     dy(1,i)=re(i) : next i
5238 zr%=ireg(2,ji) : gosub 9200 : if regart=10 then
     for i=1 to 300 : dy(2,i)=re(i) : next i :
     zrsav%=zr% : gosub 18800
5239 gosub 7500 : gosub 12000 : rem header of regr.
     menu
5240 for j=1 to 6 : fit%(j)=99 : if raster(j,ji)=0 then
     goto 5450
5245 next j : loopcount%=0 : on error goto 4250 :
     erwwflag%=0
rem ++++++++++++infinite loop begin ++++++++++++++++
5250 if b$="f" then col%=0 : row%=22 : call
     curs(col%,row%) : gosub 3645 : insav$=format$(26)
     : b$="e"
5255 if b$="i" then insav$="Interpreter on  exit=any
     softkey" : gosub 20000 : b$="e"
5260 gosub 5600 : rem setup suchgrenzen und parameter
5270 loopcount%=loopcount%+1
5280 if loopcount%>ireg(7,ji) or b$="n" then print#11,
     "regression terminated early" : b$="e" : goto 5400
     else goto 5290
5282 print#11, "fitcontrol parameter:
     ";fit%(1);fit%(2);fit%(3);fit%(4);" factors
     :";fak%(1,ji);fak%(2,ji);fak%(3,ji);fak%(4,ji)
5290 gosub 6700
5300 for j=1 to 6 : if erwwflag%=j then 5305
5302 erww(j,ji)=nvar(j+4,zr%) : next j
5305 erwwflag%=0 : gosub 5500 : rem teste Var
5310 gosub 5800 : rem auswerten der Suche
5350 if flag%=1 then goto 5400 else goto 5250
rem +++++++++++++inf loop end +++++++++++++++++++++
5400 on error goto 4200 : ivar(10,zr%)=regart
5405 gosub 9900 : if drucker%>0 then gosub 6900
5410 qr%=ireg(2,ji) : zr%=ireg(6,ji) : if regart>16
     then regart=1
5420 gosub 6150
5450 next ji
5480 row%=22 : call curs(col%,row%) : print space$(159)
5485 h1$=format$(9)+format$(10) : gosub 8700
5490 return
rem *************check variables*******************
5500 for j=1 to 6
5510 if abs(erww(j,ji))>9999 or erww(j,ji)=0 or
     fak%(j,ji)>4096 then goto 5550
5520 next j
5530 return
5550 print#11, "estimate=0 or >9999 oder factor>4096" :
     gosub 6900 : goto 5450
5600 for j=1 to 6 : rem set up search grid
     **********************************
5610 if fak%(j,ji)=-1 then lstep(j)=1 : lbeg(j)=1 :
     lend(j)=1 : goto 5730
5615 if fak%(j,ji)=0 then lstep(j)=erww(j,ji) :
     lbeg(j)=lstep(j) : lend(j)=lstep(j) : goto 5730
5640 if fit%(j)=99 then i2=3 : i3=3 : fit%(j)=0 : goto
5700
5650 fit%(j)=fit%(j)-sgn(fit%(j))
5655 if fit%(j)=0 then i2=1 : i3=1 : if
     fak%(j,ji)>=maxfak%/4 and anzan%<2 then
     fak%(j,ji)=fak%(j,ji)/2
5660 if fit%(j)=1 then i2=1 : i3=2
5665 if fit%(j)=-1 then i2=2 : i3=1
5670 if fit%(j)>=2 then i2=0 : i3=3
5675 if fit%(j)<=-2 then i2=3 : i3=0
```

```
5680 if abs(fit%(j))>=3 then fit%(j)=sgn(fit%(j))*2 :
     fak%(j,ji)=fak%(j,ji)*2
5700 lstep(j)=raster(j,ji)*fak%(j,ji)
5710 lbeg(j)=erww(j,ji)-i2*lstep(j) : lend(j)=erww
     (j,ji)+i3*lstep(j)
5725 if lbeg(j)<=0 then lbeg(j)=abs(lstep(j)/2)
5730 next j
5750 gosub 12500 : return
rem ********* Evaluate result of fit *****************
5800 maxfak%=0 : h1=0 : anzan%=0
5810 for j=1 to 6 : h2=erww(j,ji)
5820 if fak%(j,ji)=-1 or fak%(j,ji)=0 then goto 5920
5825 if fit%(j)<>0 then h1=1
5830 if h2<=lbeg(j) then fit%(j)=fit%(j)-2 : h1=1 :
     anzan%=anzan%+1
5840 if h2>=lend(j) then fit%(j)=fit%(j)+2 : h1=1 :
     anzan%=anzan%+1
5850 if abs(h2)<abs(lstep(j)) and abs(fit%(j))>2 then
     fak%(j,ji)=fak%(j,ji)/2 : fit%(j)=fit%(j)-
     sgn(fit%(j))
5860 if fak%(j,ji)>maxfak% then maxfak%=fak%(j,ji)
5870 if fit%(j)<>0 then h1=1
5920 next j
5930 if maxfak%<2 and h1=0 then flag%=1 else flag%=0
5990 gosub 12700 : return
6000 gosub 4900 : rem
***********************Simulation*****************
6030 input "Please enter type of simulation, and
     register eg:1,86";regart,zr% : n$="m" : if
     regart>20 or regart<1 or zr%<1 or zr%>99 then
     gosub 3395 : goto 500
6035 call curs(col%,row%) : print space$(239) : print
     space$(159) : call curs(col%,row%)
```

```
6040 for ji=1 to 6 : call curs(col%,row%) :
     h1$=mid$(filename$(ji),regart*4-3,4) : if h1$="
     " then 6048
6042 ileer=10 : prompt$="please enter "+h1$+" " :
     in$=str$(nvar(4+ji,zr%)) : gosub 40000 :
     h1=val(in$) : nvar(4+ji,zr%)=h1
6044 call curs(col%,row%) : print space$(79)
6046 next ji
6048 call curs(col%,row%) : print space$(79)
6050 gosub 9900 : rem speichere Var
6100 gosub 9200 : rem get Bindung in Register
6120 qr%=zr% : call curs(col%,row%) : b$="" : input
     "destination register (1-99) ";zr% : call
     curs(col%,row%) : print space$(79) : if zr%>100 or
     zr%<1 then goto 6120
6150 rem on error goto 4500 Sprungadresse
6202 for i=1 to 10 : nvar(i,zr%)=nvar(i,qr%) :
     ivar(i,zr%)=ivar(i,qr%) : next i
6204 nvar(3,zr%)=nvar(5,zr%) : ivar(10,zr%)=regart
6206 if nvar(6,zr%)=0 or nvar(8,zr%)=0 or nvar(5,zr%)=0
     or nvar(1,zr%)=0 then gosub 3395 : goto 500
6207 p1=nvar(5,zr%) : p2=nvar(6,zr%) : p3=nvar(7,zr%) :
     p4=nvar(8,zr%) : p5=nvar(9,zr%) : p6=nvar(10,zr%)
6210 if ivar(5,zr%)<21 then  gosub 7400
6215 if regart=10 then gosub 18800 : gosub 18000 : goto
6250
6220 for i=1 to ivar(5,zr%)+1 : Ligandkonz=re(i+100) :
     S=re(i+100)
6230 Enzkonz=nvar(1,zr%) : gosub 6500
6240 re(i)=signal : next i
6250 ivar(3,zr%)=1 : ivar(4,zr%)=ivar(5,zr%)+1 : row%=4
     : call curs(col%,row%)
```

```
6265 n$="Sim."+str$(ivar(10,zr%))+" SdR"+str$(stdevreg)
     +" SR    " : m$=str$(qr%)+"   " : hi$=mid$
     (n$,1,18)+mid$(m$,2,2)
6270 n$=mid$(datst$,11,10)+"    " : m$=mid$(vat$,61,
     17)+" K:"+str$(rootsq)+space$(12) : hi$=hi$+
     mid$(n$,1,10)+mid$(m$,1,29)+"R"
6275 gosub 9910 : gosub 9100 : gosub 3300
6295 return
6500 rem ********* calculate signal********************
6510 if S=0 or p2<0 or p3<0 or p4<0 or p1<0 then
     signal=0 : return
6520 on regart gosub
6530,6540,6550,6560,6570,6580,6590,6600,6610,6620,6630,
6640,6650,6660,6670,6680
6525 return
6530 signal=(p1/(1+p2/S)) : return
6540 signal=(p1*S+p2*S^2)/(1+p3*S+p4*S^2) : return
6550 signal=p1*S/(p2+S)+p3*S/(p4+S) : return
6560 if p3<=0 or p3>5 then signal=0 : return else
     signal=(p1*S^p3/(p2^p3+S^p3)) : return
6570 signal=p1*((S/p2+S^2/p2)/(1+2*S/p2+S^2/p2)) :
     return
6580 signal=p1*((S/p2+S^2/(p3*p2))/(1+2*S/p2+S^2/
     (p3*p2))) : return
6590 signal=p1-p3*S/(p2+S) : return
6600 signal=p1*S/(p2+S*(1+p4/p3)) : return
6610 signal=p1*S/(p2*(1+p4/p3)+S*(1+p4/p3)) : return
6620 return : rem 10
6630 signal=((p1+p2*p4*S)*S)/(1+p2*S+p2*p3*S^2) :
     return
6640 signal=(p1+p4+(p2*p4+p3*p1)*S)*S/(1+(p2+p3)*
     S+p2*p3*S^2) : return
6650 signal=(p1+p2*p4*S)*S/(1+p2*S+p2*p4*S^2) : return
```

```
6660 signal=p1+p1*p3*p4*S/(1+(p2+p3)*S+p2*p3*S^2) :
     return
6670 signal=p1-((p1-p3)/(1+S/p2)) : return
6680 signal=p1+p2*S : return
rem ********** direct curve fit subroutine ***********
6700 dev=999 : cycle=0 : Enzkonz=nvar(1,zr%) : tstern=0
     : tcycle=0 : h1=1
6702 for tt=1 to 4 : h1=h1*((lend(tt)-lbeg(tt))/lstep
     (tt)+1)
6703 next tt : row%=35 : col%=0 : call curs(col%,row%)
     : print space$(80) : total=h1*10
6705 for p1=lbeg(1) to lend(1)+lstep(1)/1000 step
     lstep(1)
6710  for p2=lbeg(2) to lend(2)+lstep(2)/1000 step
     lstep(2)
6715   for p3=lbeg(3) to lend(3)+lstep(3)/1000 step
     lstep(3)
6720    for p4=lbeg(4) to lend(4)+lstep(4)/1000 step
     lstep(4) : tcycle=tcycle+1
6722      tt=tcycle/total*800 : if tt>tstern and tt<81
     then row%=35 : col%=tstern : call curs(col%,row%)
     : print string$(tt-tstern,42) : tstern=tt
6723      a$=inkey$ : if len(a$)=1 then gosub 7800
6725      for p5=lbeg(5) to lend(5) step lstep(5)
6730       for p6=lbeg(6) to lend(6) step lstep(6)
6735          devm=0 : i1=0
6740          if regart=10 then gosub 19000 : goto 6765
6750          for i=von%+1 to bis%+1
6755           i1=i1+1 : S=re(i+100) : gosub 6500 :
     devm=devm+(signal-re(i))^2*dy(1,i)
6760          next i : devm=(devm/i1)^.5 : if devm<dev
     then nvar(5,zr%)=p1 : nvar(6,zr%)=p2 :
     nvar(7,zr%)=p3 : nvar(8,zr%)=p4 : nvar(9,zr%)=p5 :
     nvar(10,zr%)=p6 : dev=devm
```

```
6765        next p6
6770       next p5
6775     next p4
6780    next p3
6782   next p2
6784 next p1
6790 rootsq=dev
6797 return : rem goto 6900
rem *****************print result *********************
6900 on error goto 7750
6905 print #11, : print #11, : on error goto 6995
6910 print#11,"Curve fit to register :   ";str$(zr%);"
     ";mid$(vat$,61,20);"   ";mid$(datst$,1,20)
6920 print#11, using pres$;mid$(vat$,91,20),nvar
     (1,zr%),mid$(vat$,111,18),nvar(2,zr%)
6925 print#11, using pres2$;von%,bis%,stdevreg
6930 if regart<10 then print#11,pres3$;mid$
     (remod$,regart*20-19,20)
6935 if regart=10 then print#11,pres3$;format$(26)
6940 if regart>10 then print#11,pres3$;mid$
     (remoe$,(regart-10)*20-19,20)
6945 h1$="Result     :  "
6950 for i=1 to 6 : if i=3 and iparm<3 then 6965
6952 h1$=h1$+"  "+mid$(filename$(i),regart*4-3,4)+" ="
6955 if abs(nvar(4+i,zr%))>9999 or abs(nvar(4+i,
     zr%))<0.001 then h1$=h1$+exp$ else
     h1$=h1$+"#####.####"
6960 if i=3 and iparm>2 then h1$=h1$+space$(80-
     len(h1$))
6962 next i
6965 h1$=h1$+" mean root sq.=" : if abs(rootsq)<0.01
     then h1$=h1$+exp$ : goto 6970 else h1$=h1$+
     "#####.#####" : goto 6970
6970 if len(h1$)<160 then h1$=h1$+space$(160-len(h1$))
```

```
6975 if iparm<3 then print#11, using mid$(h1$,1,80);
     nvar(5,zr%),nvar(6,zr%),rootsq
6980 if iparm>2 then print#11, using mid$(h1$,1,80);
     nvar(5,zr%),nvar(6,zr%),nvar(7,zr%) : print#11,
     using mid$(h1$,81,80);nvar(8,zr%),
     nvar(9,zr%),nvar(10,zr%),rootsq
6985 print#11, : print#11,
6990 on error goto 4250 : return
6995 resume 6990
rem
*************input***********************************
7000 col%=0 : row%=23 : call curs(col%,row%) : print
     space$(239) : call curs(col%,row%)
7010 input "select parameter to change (1-6)";i4 : if
     i4>6 or i4<1 then 7000
7020 call curs(col%,row%) : print space$(159) : call
     curs(col%,row%)
7030 if i3=2 then print fak%(i4,ji);" searchfactor  ";
     : input "new";fak%(i4,ji)
7032 if i3=3 then print erww(i4,ji);" estimate      ";
     : input "new";erww(i4,ji) : erwwflag%=i4
7034 if i3=2 then print raster(i4,ji);" precision  "; :
     input "new";raster(i4,ji)
7036 if i3=1 then print fit%(i4);"  fitcontrol     "; :
     input "new";fit%(i4)
7090 return
rem ******************input 3 variables**************
7100 ileer=20 : gosub 40000 : for i=1 to 5 : ik(i)=0 :
     next i : if mid$(in$,1,1)="e" then return
7105 ikomm=0 : il=len(in$) : for i=1 to il : if
     mid$(in$,i,1)="," then ikomm=ikomm+1 :
     ik(ikomm+1)=i
7110 next i : ik(ikomm+2)=i
```

```
7120 if ik(2)-ik(1)>0 then h1=val(mid$(in$,ik(1)+1,
     ik(2)-ik(1)-1)) else h1=0
7130 if ik(3)-ik(2)>0 then h2=val(mid$(in$,ik(2)+1,
     ik(3)-ik(2)-1)) else h2=0
7140 if ik(4)-ik(3)>0 then h3=val(mid$(in$,ik(3)+1,
     ik(4)-ik(3)-1)) else h3=0
7145 if ikomm>3 or h1=0 then h1=1 : h2=1 : h3=-1 :
     return
7150 if ikomm>0 then 7160
7152 h1#=1E-20 : for i=1 to 40 : h1#=h1#*10 : if
     h1#>abs(h1/1000) then 7156
7154 next i : h1=1 : h2=1 : h3=-1 : return
7156 if h1#>abs(h1/500) then h1#=h1#/2
7158 h2=h1# : h3=128 : return
7160 if ikomm>1 then 7170
7162 if h2=0 then h2=h1/10 : h3=0 : return
7164 if h2=-1 then h2=h1/10 : h3=-1 : return
7170 if h2<0 then h2=1 : h3=-1 : return
7175 if h2=0 then h2=h1/10 : h3=0 : return
7190 return
7400 rem *************expand*************************
7410 for i=1 to ivar(5,zr%)+1 : dx(1,i)=re(100+i) :
     next i : dx(1,ivar(5,zr%)+2)=re(ivar
     (5,zr%)+101)*1.2 : w=0
7415 for i=1 to ivar(5,zr%)+1 : w=w+1 : re(w+100)=
     dx(1,i) : if i=1 then w=w-1
7420 w=w+1 : re(w+100)=dx(1,i)+(dx(1,i+1)-dx(1,i))/3
7425 w=w+1 : re(w+100)=dx(1,i)+(dx(1,i+1)-dx(1,i))/3*2
7430 next i : ivar(5,zr%)=w-2
7445 return
rem ************** weight for weighted fit ***********
7500 if stdevreg=0 then goto 7590
7520 h1=0 : h2=0
```

```
7550 for i=von%+1 to bis%+1 : if dy(1,i)=0 then goto
7560 else h1=h1+abs(re(i)/dy(1,i)) : h2=h2+1
7560 next i : h1=h1/h2
7570 for i=von%+1 to bis%+1 : if h1=0 or dy(1,i)=0 then
     dy(1,i)=1 else : h2=abs(re(i)/dy(1,i)) :
     dy(1,i)=h2/h1
7580 next i
7590 return
rem *************** Time and date**********************
7600 minuten%=0 : sekunden%=0 : call zeit(minuten%,
     sekunden%) : stunden%=cint(minuten%/256) :
     minuten%=minuten% mod 256 : sekunden%=sekunden
     %/256
7620 i2=int(i1/3600) : i3=int((i1-3600*i2)/60) : i4=i1-
     3600*i2-60*i3 : sekunden%=sekunden%+i4 : if
     sekunden%>59 then minuten%=minuten%+1 :
     sekunden%=sekunden%-60
7630 minuten%=minuten%+i3 : if minuten%>59 then
     minuten%=minuten%-60 : stunden%=stunden%+1
7640 rem call datum(jahr%,tag%,wtag%) : monat%=tag%/256
     : tag%=tag% mod 256
7660 rem stunden%=stunden%+i2 : if stunden%>23 then
     stunden%=stunden%-24 : tag%=tag%+1
7690 return
rem *****************printer off**********************
7700 resume 7710
7710 col%=0 : row%=22 : call curs(col%,row%) : print
     "printer not responding (e=exit r=retry) :
     a$=input$(1)
7720 if a$="r" then 7811 else drucker%=0 : goto 7814
7750 resume 7760
7760 col%=0 : row%=22 : call curs(col%,row%) : print
     "printer not responding (e=exit r=retry) :
     a$=input$(1)
```

```
7770 if a$="r" then 6900 else drucker%=0 : goto 6990
rem
*****************softkeys****************************
7800 tsoft=asc(a$)-17 : if tsoft<1 or tsoft>8 then 7900
7805 on tsoft gosub
7810,7820,7830,7840,7850,7860,7870,7880
7807 return
7810 drucker%=drucker%+1 : if drucker%>2 then
     drucker%=0
7811 on error goto 7700
7812 if drucker%>0 then print#11,
7813 if drucker%=2 then h1$="" : for i=1 to 6 :
     h1$=h1$+"     "+mid$(filename$(i),regart*4-3,4)+"
     " : next i : h1$=h1$+"mean root sq." :
     print#11,h1$
7814 on error goto 4250
7815 col%=75 : row%=34 : call curs(col%,row%) : print
     drucker% : col%=0 : return
7820 return
7830 i3=2 : gosub 7000 : return
7840 i3=3 : gosub 7000 : return
7850 i3=4 : gosub 7000 : return
7860 i3=1 : gosub 7000 : return
7870 b$="n" : col%=0 : row%=23 : call curs(col%,row%) :
     print "Please wait for termination of fit in
     process" : return
7880 col%=0 : row%=0 : call curs(col%,row%)
7885 h1$=format$(9)+format$(10) : gosub 8700 : goto 500
7900 if a$="&" or a$="i" then b$="i" : col%=0 : row%=23
     : call curs(col%,row%) : print "Please wait for
     termination of fit in process" : return
7910 if a$="f" then b$="f" : col%=0 : row%=23 : call
     curs(col%,row%) : print "Please wait for
     termination of fit in process" : return
```

```
7980 return
rem **************UPPERCASE***************************
7990 il=len(in$) : for i=1 to il : ia=asc(mid$(in
     $,i,1)) : if ia>96 and ia<123 then ia=ia-32 :
     mid$(in$,i,1)=chr$(ia)
7995 next i : return
rem
*****************Keylabel*****************************
8700 if len(h1$)<160 then h1$=h1$+space$(161-len(h1$))
8705 for i=1 to 8 : m$=str$(i) : h2$=chr$(27)+"&f0a"
     +mid$(m$,2,1)+"k16d1L"+mid$(h1$,i*18-15,16)+chr$
     (17+i) : print h2$; : next i
8710 print chr$(27); "&jB"; : return
9100 rem ************ write re () to workfile  ********
9105 if zr%>100 or zr%<1 then zr%=1
9110 n$=space$(253)
9120 for wl=1 to 63 : mid$(n$,wl*4-3,4)=mks$(re(wl)) :
     next wl : lset e1$=n$
9130 for wl=101 to 163 : mid$(n$,(wl-100)*4-
     3,4)=mks$(re(wl)) : next wl
9135 lset e2$=n$ : put #9,zr%
9140 return
9200 rem ********* read re () from workfile ***********
9205 if zr%>100 or zr%<1 then zr%=1
9210 get #9,zr%
9220 for wl=1 to 64 : re(wl)=0 : next wl
9230 for wl=1 to 63 : re(wl)=cvs(mid$(e1$,wl*4-3,4)) :
     next wl
9235 for wl=101 to 163 : re(wl)=cvs(mid$(e2$,(wl-
     100)*4-3,4)) : next wl
9240 return
9800 rem ****** read variables from workfile **********
9802 if zr%>100 or zr%<1 then zr%=1
9805 get #8,zr%
```

```
9810 for wl=1 to 10
9820 ivar(wl,zr%)=cvi(mid$(vat$,wl*2-1,2)) :
     nvar(wl,zr%)=cvs(mid$(vat$,wl*4+17,4))
9830 next wl : return
9900 rem ****** write variables to workfile ***********
9902 if zr%>100 or zr%<1 then zr%=1
9905 get #8,zr% : hi$=mid$(vat$,61,70)+"          " :
9910 n$=space$(130)
9915 for wl=1 to 10
9930 mid$(n$,wl*2-1,2)=mki$(ivar(wl,zr%))
9940 mid$(n$,wl*4+17,4)=mks$(nvar(wl,zr%))
9945 next wl : mid$(n$,61,70)=mid$(hi$,1,70) : lset
     vat$=n$ : put #8,zr%
rem ****************set update pointers****************
9950 iu(zr%)=1 : pointer%(13)=0
9955 for wl=1 to 5 : if tque(wl+offo%)=zr% then
     ip(27+wl)=1 : pointer%(12)=0
9960 next wl
9965 for wl=1 to 10 : if tque(wl)=zr% then ip(wl+40)=1
     : pointer%(12)=0
9970 next wl : return
rem ****************linear regression *****************
10000 row%=22 : call curs(col%,row%) : print
     space$(239) : print space$(159) : call
     curs(col%,row%) : iparm=2
10002 input "Please enter register number and points
     (from/to)";zr%,von%,bis% : if zr%<1 or zr%>99 then
     return
10005 call curs(col%,row%) ; print space$(239) : call
     curs(col%,row%) : Print "Please select
     transformation before regression"
10006 print "1=No transformation 2=double reciprocal
     3=Eadie Hofstee 4=Hannes Woolfe" : input i : if
     i>1 and i<5 then goto 10500
```

```
10007 gosub 9200
10010 i1=0 : sux=0 : suy=0 : suxy=0 : suyy=0 : suxx=0 :
     nvar(6,zr%)=0 : nvar(7,zr%)=0
10020 for i=von% to bis% : ywert=re(i) :
     xwert=re(i+100)
10050 i1=i1+1 : sux=sux+xwert : suy=suy+ywert :
     suxy=suxy+xwert*ywert : suyy=suyy+ywert^2 :
     suxx=suxx+xwert^2
10060 next i : if i1=0 then goto 10065 else
     roy=suyy/i1-(suy/i1)^2 : rox=suxx/i1-(sux/i1)^2
10065 if rox=0 or roy=0 or i1=0 then print "Invalid
     linear regression" : for k=1 to 4000 : i=k : next
     k : goto 10250
10070 Steigung=(suxy/i1-sux/i1*suy/i1)/rox
10080 rootsq=Steigung*rox^.5/roy^.5 : Schnittp=suy/i1-
     Steigung*sux/i1
10100 print#11,"Linear regression :  Register ";zr%;"
     from ";von%;" to ";bis%
10120 print#11,"Result :  Intercept: ";Schnittp;"
     Slope: ";Steigung;" Correlation: ";rootsq
10140 row%=23 : call curs(col%,row%) : print "Result :
     Intercept: ";Schnittp;" Slope: ";Steigung;"
     Correlation: ";rootsq
10160 nvar(5,zr%)=Schnittp : nvar(6,zr%)=Steigung :
     nvar(7,zr%)=1 : nvar(8,zr%)=1 : gosub 9900
10200 qr%=zr% : row%=24 : call curs(col%,row%) : input
     "Please enter destination of regression (1 to
     99)";zr% : if zr%>99 or zr%<1 then return
10220 regart=16 : gosub 6150
10250 row%=22 : call curs(col%,row%) : print
     space$(239) : print space$(159)
10300 return
rem 10100 if zr%<1 or zr%>99 then return
rem 10202 for i=1 to 10 : nvar(i,zr%)=nvar(i,qr%) :
     ivar(i,zr%)=ivar(i,qr%) : next i
rem 10204 nvar(3,zr%)=nvar(5,zr%)
rem 10210 rem if ivar(5,zr%)<21 then  gosub 7400
rem 10220 for i=1 to ivar(5,zr%)+1 : xwert=re(i+100)
rem 10230   re(i)=Schnittp+xwert*Steigung
rem 10240 next i
rem 10250 ivar(3,zr%)=1 : ivar(4,zr%)=ivar(5,zr%)+1
rem 10265 n$="Linear Regr. Source " : m$=str$(qr%)+"  "
     : hi$=mid$(n$,1,18)+mid$(m$,2,2)
rem 10270 n$=mid$(datst$,11,10)+"   " :
     m$=mid$(vat$,61,17)+"K: "+str$(rootsq)+"        "
     : hi$=hi$+mid$(n$,1,10)+mid$(m$,1,39)+"R"
rem 10275 gosub 9910 : gosub 9100 : gosub 3300
rem 10300 return
rem **************************************************
10500 regart=i+15 : gosub 9200 : gosub 9800
10505 i1=0 : sux=0 : suy=0 : suxy=0 : suyy=0 : suxx=0 :
     nvar(6,zr%)=0 : nvar(7,zr%)=0
10510 for i=von% to bis% : V=re(i) : S=re(i+100)
10515 if regart=17 then if V=0 or S=0 then 10560
10520 if regart=17 then xwert=1/S : ywert=1/V
10525 if regart=18 then if S=0 then 10560
10530 if regart=18 then xwert=V : ywert=V/S
10535 if regart=19 then if V=0 then 10560
10540 if regart=19 then xwert=S : ywert=S/V
10550 i1=i1+1 : sux=sux+xwert : suy=suy+ywert :
     suxy=suxy+xwert*ywert : suyy=suyy+ywert^2 :
     suxx=suxx+xwert^2
10560 next i : if i1=0 then goto 10565 else
     roy=suyy/i1-(suy/i1)^2 : rox=suxx/i1-(sux/i1)^2
10565 if rox=0 or roy=0 or i1=0 then print "Invalid
     linear regression" : for k=1 to 4000 : i=k : next
     k : goto 10940
```

```
10570 Steigung=(suxy/i1-sux/i1*suy/i1)/rox
10580 rootsq=Steigung*rox^.5/roy^.5 : Schnittp=suy/i1-
     Steigung*sux/i1
10600 if regart=17 then nvar(6,zr%)=abs(1/
     (Schnittp/Steigung)) : nvar(5,zr%)=abs(1/Schnittp)
10650 if regart=18 then nvar(6,zr%)=abs(1/Steigung) :
     nvar(5,zr%)=abs(Schnittp/Steigung)
10660 if regart=19 then nvar(6,zr%)=abs
     (Schnittp/Steigung) : nvar(5,zr%)=abs(1/Steigung)
10910 gosub 9900 : if drucker%>0 then gosub 6900
10920 qr%=zr% : input "Please enter destination
     register";zr% : if zr%>99 or zr%<1 then return
10930 regart=1 : gosub 6150
10940 row%=22 : call curs(col%,row%) : print
     space$(239) : print space$(159)
10950 return
rem ***************** wait for input *****************
11000 print chr$(27);"-z2N";
11010 a$=input$(1) : b=asc(a$)-89
11020 print chr$(27);"-z0N"; : rem off keys
     ************************
11025 iflag=1 : tsoft=0 : if asc(a$)>17 and asc(a$)<26
     then tsoft=asc(a$)-17 : return
11030 if a$="*" then pinit=1 : forts=1 : running%=0 :
     return
11040 if a$="/" then pinit=1 : forts=2 : running%=0 :
     return
11050 if a$="+" then pinit=1 : forts=3 : running%=0 :
     return
11090 iflag=1 : return
rem ***************direct curve fit menu **************
12000 col%=0 : row%=44 : call curs(col%,row%) : print
     space$(79) : gosub 9800
```

```
12010 row%=24 : call curs(col%,row%) : print
     space$(160)
12020 row%=26 : call curs(col%,row%) : print
     "********************  D I R E C T   C U R V E
     F I T ***********************"
12030 row%=27 : call curs(col%,row%) : print space$(80)
12110 row%=28 : call curs(col%,row%) : print "source
     register    : ";str$(zr%);"
     ";mid$(vat$,61,20);"       ";mid$(datst$,1,20)
12120 row%=29 : call curs(col%,row%) : print using
     pres$;mid$(vat$,91,20),nvar(1,zr%),mid$(vat$,111,1
     8),nvar(2,zr%)
12130 row%=30 : call curs(col%,row%) : print using
     pres2$;von%,bis%,stdevreg
12135 row%=31 : call curs(col%,row%) : print
     space$(240) : call curs(col%,row%)
12140 if regart<10 then print pres3$;mid$(remod$,
     regart*20-19,20)
12150 if regart=10 then print pres3$;format$(26)
12160 if regart>10 then print pres3$;mid$(remoe$,
     (regart-10)*20-19,20)
12180 row%=32 : call curs(col%,row%) : print : print
     "batch position number ";ji
12190 col%=60 : row%=34 : call curs(col%,row%) : print
     "Printlevel" : col%=75 : call curs(col%,row%) :
     print drucker% : col%=0
12200 row%=35 : call curs(col%,row%) : print space$(80)
12220 row%=36 : call curs(col%,row%) : print "
     Param.1    Param.2    Param.3    Param.4
     Param.5    Param.6"
12250 col%=0 : row%=37 : print "Meaning         " : for
     i=1 to 6 : col%=i*11+7 : call curs(col%,row%) :
     print mid$(filename$(i),regart*4-3,4) : next i
```

```
12260 col%=0 : row%=38 : call curs(col%,row%) : print
     "Pos.Negative   " : print "Fitcontrol     " : print
     "Gridfactor  "
12270 print "Range from      " : print "Range to    " :
     print "Result         " : print "root mean sqare   "
12280 h1$="1= print   level  2=               3= input
     gridfact4= input  estimate5= input   posneg.6=
     input  controlp7=   next    fit  8=  exit     fit
     " : gosub 8700
12300 return
rem ***************************************************
12500 h1=1
12540 for i1=1 to 6 : h1=h1*((lend(i1)-lbeg(i1))/
     lstep(i1)+1) : next i1 : suchcyc=h1*ivar(5,zr%)
12545 i1=int(suchcyc/150) : if regart=10 then i1=i1*8 :
     if len(format$(26))>30 then i1=i1*2
12550 gosub 7600 : col%=0 : row%=34 : call
     curs(col%,row%) : zest$="Fit Number  ###    is
     expected at     ## : ## : ##   " : print using
     zest$;loopcount%+1,stunden%,minuten%,sekunden%
12582 col%=14 : row%=41 : call curs(col%,row%) : print
     using "+#.###^^^^ ";lbeg(1),lbeg(2),lbeg(3),
     lbeg(4),lbeg(5),lbeg(6)
12583 col%=14 : row%=42 : call curs(col%,row%) : print
     using "+#.###^^^^ ";lend(1),lend(2),lend(3),
     lend(4),lend(5),lend(6)
12587 col%=14 : row%=39 : call curs(col%,row%) : print
     using "  +###      ";fit%(1),fit%(2),fit%(3),
     fit%(4),fit%(5),fit%(6)
12589 col%=14 : row%=40 : call curs(col%,row%) : print
     using " #####      ";fak%(1,ji),fak%(2,ji),
     fak%(3,ji),fak%(4,ji),fak%(5,ji),fak%(6,ji)
12600 col%=14 : row%=38 : call curs(col%,row%) : print
     using " ##  ##     ";pne%(1),da%(1),pne%(2),
     da%(2),pne%(3),da%(3),pne%(4),da%(4),pne%(5),da%(5
     ),pne%(6),da%(6)
12690 return
12700 rem
12710 col%=14 : row%=43 : call curs(col%,row%) : print
     using "+#.###^^^^ ";nvar(5,zr%),nvar(6,zr%),
     nvar(7,zr%),nvar(8,zr%),nvar(9,zr%),nvar(10,zr%)
12720 col%=20 : row%=44 : call curs(col%,row%) : print
     rootsq
12730 if drucker%>=2 then print#11, using "+#.###^^^^
     ";nvar(5,zr%),nvar(6,zr%),nvar(7,zr%),nvar(8,zr%),
     nvar(9,zr%),nvar(10,zr%),rootsq
12790 return
rem **********interpreter simulation******************
18000 zrsav%=zr%
18020 intp$=insav$ : an%=ivar(5,zr%) : for i=1 to 10 :
     zr(i)=0 : next i
18050 gosub 18700 : for i=1 to 300 : dy(2,i)=re(i) :
     next i
18100 gosub 20010 : rem Simulation in rey(i)
18200 for i=1 to ivar(5,zr%)+1 : re(i)=rey(i) : next i
18490 return
rem **************load constants **********************
18700 c(1)=p1 : c(2)=p2 : c(3)=p3 : c(4)=p4 : c(5)=p5 :
     c(6)=p6 : return
rem
*****************************************************
18800 in$=format$(26) : il=len(in$) : for i=1 to il :
     if mid$(in$,i,2)="RR" then h1$=str$(zr%)+"   " :
     mid$(in$,i,2)=mid$(h1$,2,2)
18810 next i : insav$=in$ : return
rem **********interpreter direct curve fit ************
19000 intp$=insav$ : an%=ivar(5,zr%) : for i=1 to 10 :
     zr(i)=0 : next i
```

```
19100 gosub 18700
19250 gosub 20010
19500 rem
19760     devm=0 : i1=0
19762     for i=von%+1 to bis%+1
19764      i1=i1+1 : if re(i)<>rey(i) then
    devm=devm+(rey(i)-re(i))^2*dy(1,i)
19766     next i : devm=(devm/i1)^.5 : if devm<dev then
     nvar(5,zr%)=p1 : nvar(6,zr%)=p2 : nvar(7,zr%)=p3 :
     nvar(8,zr%)=p4 : nvar(9,zr%)=p5 : dev=devm
19900 return
rem *************interpreter*************************
20000 col%=0 : row%=23 : prompt$="" : in$=insav$ :
     ileer=75 : gosub 40000 : gosub 7990 : if ia>17 and
     ia<26 or ia=189 or in$="EXIT" then return
20005 zlev=0 : an%=0 : for i=1 to 10 : zr(i)=0 : next i
     : insav$=in$
20007 intp$=in$
20010 l=len(intp$) : if l<1 then goto 25990
20090 level=0 : lmax=0 : gpos=0 : on error goto 50000
rem  *********** analyse string **********************
20100 for i=1 to l : h$=mid$(intp$,i,1)
20110 if h$="(" then level=level+1
20130 if level<0 then goto 29900
20140 if lmax<level then lmax=level
20150 if lmax=level and h$="(" then apos=i
20155 if lmax=level and h$=")" then zpos=i
20160 if h$=")" then level=level-1
20165 if h$="=" then gpos=i
20170 if asc(h$)<30 or asc(h$)>128 then goto 29900
20190 next i : rem print l,lmax,gpos
20200 if level<>0 or gpos=0 then goto 29900
20300 if lmax=0 then h1$=mid$(intp$,gpos+1,(l-gpos)) :
     apos=gpos : zpos=l+1 : goto 20500
```

```
20400 h1$=mid$(intp$,apos+1,(zpos-apos-1))
20500 opa=0 : for i=1 to len(h1$) : h$=mid$(h1$,i,1)
20510 if h$="^" then opo=i : opa=1 : goto 21000
20520 if h$="L" then opo=i : opa=2 : goto 21000
20530 next i
20550 for i=1 to len(h1$) : h$=mid$(h1$,i,1)
20560 if h$="*" then opo=i : opa=3 : goto 21000
20570 if h$="/" then opo=i : opa=4 : goto 21000
20580 if h$="D" then opo=i : opa=5 : goto 21000
20590 next i
20600 for i=1 to len(h1$) : h$=mid$(h1$,i,1)
20610 if h$="+" then opo=i : opa=7 : goto 21000
20620 if h$="-" then opo=i : opa=8 : goto 21000
20630 next i
20700 if lmax=0 then goto 25500
20710 if opa=0 then h2$=mid$(intp$,1,apos-1) :
     h3$=mid$(intp$,zpos+1,l-zpos-1) : intp$=h2$+
     h1$+h3$ : goto 20010
20720 goto 29800
21000 lpo=0 : rpo=0 : lrega=0 : if opo=1 then goto
21050
21010 for i=opo-1 to 1 step-1 : h$=mid$(h1$,i,1)
21020 if h$="^" or h1$="L" or h$="*" or h$="/" or
     h$="+" or h$="-" or h$="D" then lpo=i : goto 21030
21025 next i : lpo=0
21030 for i=lpo+1 to opo-1 : h$=mid$(h1$,i,1)
21035 if lrega=0 then gosub 27000 : lque=ique :
     lrega=rega : lque2=nque2 : if rega<>0 then goto
21050
21040 next i
21050 rrega=0
21055 for i=opo+1 to len(h1$) : h$=mid$(h1$,i,1) : if
     opo=len(h1$) then rpo=len(h1$)+1 : goto 21100
```

```
21060 if h$="^" or h1$="L" or h$="*" or h$="/" or
     h$="+" or h$="-" or h$="D" then rpo=i : goto 21070
21065 next i : rpo=len(h1$)+1
21070 for i=opo+1 to rpo-1 : h$=mid$(h1$,i,1)
21075 if rrega=0 then gosub 27000 : rque=ique :
     rrega=rega : rque2=nque2 : if rega<>0 then goto
     21100
21080 next i
21100 h9$=space$(79) : mid$(h9$,lpo+1,1)=mid$(str$
     (lrega),2,1) : mid$(h9$,opo+1,1)=mid$
     (str$(rrega),2,1)
21110 mid$(h9$,opo,1)=mid$(str$(opa),2,1) :
     mid$(h9$,opo+3,1)=mid$(str$(rque),2,1)
21120 mid$(h9$,lpo+3,1)=mid$(str$(lque),2,1) : rem
     print h9$
22000 gosub 26000 : if an%<0 then an%=0
22300 gosub 23000 : if an%<0 then an%=0
22500 if lpo=0 then h2$=" " else h2$=mid$(h1$,1,lpo)
22502 if rpo=len(h1$)+1 then h4$=" " else
     h4$=mid$(h1$,rpo,len(h1$)-rpo+1)
22504 h1b$=left$(intp$,apos) : h5$=mid$(intp$,zpos,l-
     zpos+1) : h3$="Z"+str$(zlev)
22510 if lpo=0 and rpo=len(h1$)+1 and lmax>0 then
     intp$=mid$(h1b$,1,len(h1b$)-1)+h3$+mid$(h5$,2,
     len(h5$)-1) : goto 22550
22515 if lpo=0 and rpo=len(h1$)+1 and lmax=0 then
     intp$=mid$(h1b$,1,len(h1b$))+h3$+mid$(h5$,2,len(h5
     $)-1) : goto 22550
22520 intp$=h1b$+h2$+h3$+h4$+h5$
22550 goto 20010 : rem print "Neuer String ";intp$ :
goto 20010
rem  ****** perform operation on accumulator **********
23000 for i=9  to 2 step-1 : if zr(i)=0 then zlev=i :
     zr(zlev)=an%+1 : goto 23050 : REM  N+1 !!!!
```

```
23010 next i
23050 on opa goto
23100,23200,23300,23400,23500,23600,23700,23800,23900
23080 for i=1 to an%+1 : dx(zlev,i)=rex(i) : next i :
     goto 24000
23100 for i=1 to an%+1 : if rex(i)<=0 then dx(zlev,i)=1
     else dx(zlev,i)=rex(i)^rey(i)
23110 next i : goto 24000
23200 for i=1 to an%+1 : if rex(i)<=0 or rey(i)=0 then
     dx(zlev,i)=1 else dx(zlev,i)=log
     (rey(i))/log(rex(i))
23210 next i : goto 24000
23300 for i=1 to an%+1 : dx(zlev,i)=rex(i)*rey(i) :
     next i : goto 24000
23400 for i=1 to an%+1 : if rey(i)=0 then
     dx(zlev,i)=9.99E+20 else dx(zlev,i)=rex(i)/rey(i)
23420 next i : goto 24000
23500 dx(zlev,1)=((rex(1)+rex(2))/2-(3*rex(1)-
     rex(2))/2)/nvar(5,zr%)
23510 for i=2 to an% : h1=(rex(i-1)+rex(i))/2 :
     h2=(rex(i)+rex(i+1))/2
23520  dx(zlev,i)=(h2-h1)/nvar(5,zr%) : next i
23530 dx(zlev,an%+1)=(rex(an%+1)-
     (rex(an%+1)+rex(an%))/2)/nvar(5,zr%) : goto 24000
23600 rem
23700 for i=1 to an%+1 : dx(zlev,i)=rex(i)+rey(i) :
     next i : goto 24000
23800 for i=1 to an%+1 : dx(zlev,i)=rex(i)-rey(i) :
     next i : goto 24000
23900 rem
24000 rem print "zwischenresultat in register ";zlev;"
     : ";an%+1;" Punkte"  : for i=1 to an%+1 : print
     dx(zlev,i) : next i
24010 return
```

```
25500 rem ************* ende op **********************
25505 h1$=intp$ : rrega=0
25510 for i=gpos to len(h1$) : h$=mid$(h1$,i,1)
25520 if rrega=0 then gosub 27000  : rque=ique :
     rrega=rega : rque2=nque2
25530 next i : opo=gpos : rpo=len(h1$)+1
25540 lrega=7 : gosub 26000 : lrega=0
25550 for i=1 to gpos-1 : h$=mid$(h1$,i,1)
25560 if lrega=0 then gosub 27000 : lque=ique :
     lrega=rega : lque2=nque2
25565 if h$="R" then return
25580 next i
25620 if lrega<3 or lrega>10 then goto 25990 else print
     in$+space$(79-len(in$))
25630 on lrega goto
25800,25800,25650,25700,25750,25800,25850,25900,25960,
     25950
25650 print "X-Register";lque;" = ";an%;" Punkte" :
     zr%=lque : gosub 9200
25652 for i=1 to 10 : nvar(i,lque)=nvar(i,lzr%) :
     ivar(i,lque)=ivar(i,lzr%) : next i :
     hi$="Res."+str$(lzr%)+space$(30)
25654 ivar(5,zr%)=an% : gosub 9910
25656 for i=1 to an% : re(i+100)=rey(i) : next i :
     gosub 9100
25660 goto 25990
25700 print "Y-Register";lque;" = ";an%;" Punkte" :
     zr%=lque
25702 for i=1 to 10 : nvar(i,lque)=nvar(i,lzr%) :
     ivar(i,lque)=ivar(i,lzr%) : next i :
     hi$="Res."+str$(lzr%)+space$(30)
25704 gosub 9910
```

```
25706 for i=1 to an% : re(i)=rey(i) : next i : gosub
     9100
25710 goto 25990
25750 c(lque)=rey(1) : print "Konstante ";lque;" =
     ";rey(1) : goto 25990
25800 if lque<11 then ivar(lque,lque2)=rey(1) : gosub
     9900 : print "Variable ";lque;",";lque2;" =
     ";rey(1) : goto 25990
25810 if lque>10 then nvar(lque-10,lque2)=rey(1) :
     gosub 9900 : print "Variable ";lque;",";lque2;" =
     ";rey(1) : goto 25990
25820 goto 25990
25850 if an%=0 then dx(j,3)=rey(1) : dmax(j)=3 : goto
25990 : print "1 Punkt"
25855 i1=2 : for i=ivar(3,lzr%)+1 to ivar(4,lzr%)+1 :
     i1=i1+1 : dx(j,i1)=rey(i-1) : next i : dmax(j)=i1
     : print dmax(j);" Punkte eingel"
25860 goto 25990
25900 if an%=0 then dy(j,3)=rey(1) : dmax(j)=3 : goto
25990 : print "1 Punkt"
25905 i1=2 : for i=ivar(3,lzr%)+1 to ivar(4,lzr%)+1 :
     i1=i1+1 : dy(j,i1)=rey(i-1) : next i : dmax(j)=i1
     : print dmax(j);" Punkte eingel"
25910 goto 25990
25950 zr%=lque : gosub 9200 : print "Y-Wert
     ";lque;",";lque2;" = ";rey(1),lque2 : if lque2<1
     or lque2>64 then 25990
25954 re(lque2)=rey(1) : gosub 9100 : goto 25990
25960 zr%=lque : gosub 9200 : print "Y-Wert
     ";lque;",";lque2;" = ";rey(1),lque2 : if lque2<1
     or lque2>64 then 25990
25964 re(lque2+100)=rey(1) : gosub 9100 : goto 25990
25990 return
rem *************load accumulator ********************
```

```
26000 on error goto 50000
26050 on lrega goto
26070,26080,26100,26110,26120,26130,26200,26200,26150,
     26160
26060 for i=1 to an%+1 : if opa=3 or opa=4 then
     rex(i)=1 else rex(i)=0
26062 if opa=2 then rex(i)=2.71828182
26065 next i : goto 26200
26070 hx$=mid$(h1$,lpo+1,opo-lpo-1)
26075 for i=1 to an%+1 : rex(i)=val(hx$) : next i :
     goto 26200
26080 if zr(lque)<an%+1 then an%=zr(lque)-1
26085 for i=1 to an%+1 : if zr(lque)=1 then i1=1 else
     i1=i
26090 rex(i)=dx(lque,i1)
26095 next i : zr(lque)=0 : goto 26200
26100 ziel=1 : zr%=lque : gosub 27200 : goto 26200 :
     rem get X
26110 ziel=1 : zr%=lque : gosub 27300 : goto 26200 :
     rem get Y
26120 for i=1 to an%+1 : rex(i)=c(lque) : next i : goto
     26200
26130 if lque<11 then for i=1 to an%+1 :
     rex(i)=ivar(lque,lque2) : next i : goto 26200
26140 if lque>10 then for i=1 to an%+1 :
     rex(i)=nvar(lque-10,lque2) : next i : goto 26200
26150 zr%=lque : an%=0 : rex(1)=lque2 : print rex(1) :
     goto 26200
26160 ziel=1 : zr%=lque : gosub 27300 : an%=0
26162 if i<1 or lque2>nvar(5,zr%) then rex(1)=0 : print
     "out of range" : goto 26200
26165 rex(1)=rex(lque2) : print rex(1) : goto 26200
```

```
26200 nlinks%=an% : on rrega goto
26215,26220,26240,26250,26260,26270,26300,26300,26290,2
     6295
26210 for i=1 to an%+1 : rey(i)=0 : next i : goto 26300
26215 hy$=mid$(h1$,opo+1,rpo-opo-1) : for i=1 to an%+1
     : rey(i)=val(hy$) : next i : goto 26300
26220 if zr(rque)<an%+1 then an%=zr(rque)-1
26225 for i=1 to an%+1 : if zr(rque)=1 then i1=1 else
     i1=i
26230 rey(i)=dx(rque,i1)
26235 next i : zr(rque)=0 : goto 26300
26240 ziel=2 : zr%=rque : gosub 27200 : goto 26300
26250 ziel=2 : zr%=rque : gosub 27300 : goto 26300
26260 for i=1 to an%+1 : rey(i)=c(rque) : next i : goto
26300
26270 if rque<11 then for i=1 to an%+1 :
     rey(i)=ivar(rque,rque2) : next i : goto 26300
26280 if rque>10 then for i=1 to an%+1 :
     rey(i)=nvar(rque-10,rque2) : next i : goto 26300
26290 zr%=lque : an%=0 : rey(1)=rque2 : goto 26300
26295 ziel=2 : zr%=lque : gosub 27300 : an%=0
26296 if i<1 or rque2>nvar(5,zr%) then rey(1)=0 : print
     "out of range" : goto 26300
26297 rey(1)=rey(lque2) : goto 26300
26300 nrechts%=an% : rem print "akku a = ";rex(1);"
     akku b = ";rey(1)
26310 if nlinks%=nrechts% then goto 26360
26320 if nlinks%=0 then for i=1 to nrechts%+1 :
     rex(i)=rex(1) : next i : nlinks%=nrechts% :
     an%=nlinks%
26330 if nrechts%=0 then for i=1 to nlinks%+1 :
     rey(i)=rey(1) : next i : nrechts%=nlinks% :
     an%=nrechts%
26340 if nlinks%<nrechts% then an%=nrechts%
```

```
26350 if nrechts%<nlinks% then an%=nlinks%
26360 return
27000 on error goto 50000 : rem ************* type of
      source ***********
27010 rega=0 : ique=0 : nque2=0 : leseflag%=0
27015 if h$="Z" then rega=2 : ique=val(mid$(h1$,i+1,2))
      : return
27020 if h$="X" then rega=3 : ique=val(mid$(h1$,i+1,2))
      : return
27025 if h$="S" then rega=3 : ique=zrsav% : leseflag%=1
      : return
27030 if h$="Y" then rega=4 : ique=val(mid$(h1$,i+1,2))
      : return
27035 if h$="T" then rega=4 : ique=zrsav% : leseflag%=1
      : return
27040 if h$="C" then rega=5 : ique=val(mid$(h1$,i+1,2))
      : return
27045 if h$="V" then rega=6 : ique=val(mid$(h1$,i+1,2))
      : nque2=val(mid$(h1$,i+4,2)) : return
27050 if h$="A" then rega=7 : ique=val(mid$(h1$,i+1,2))
      : return
27055 if h$="B" then rega=8 : ique=val(mid$(h1$,i+1,2))
      : return
27060 if h$="N" then rega=9 : ique=val(mid$(h1$,i+1,2))
      : nque2=val(mid$(h1$,i+4,6)) : return
27070 if h$="M" then rega=10 : ique=val(mid$(h1
      $,i+1,2)) : nque2=val(mid$(h1$,i+4,6)) : return
27080 if h$="1" or h$="2" or h$="3" or h$="4" or h$="5"
      or h$="6" or h$="7" or h$="8" or h$="9" or h$="0"
      then rega=1 : return
27090 return
27200 lzr%=zr% : if leseflag%=1 then goto 27280
27210 an%=ivar(5,zr%) : gosub 9200 : rem
      ****************** get xreg******************
```

```
27250 if ziel=1 then for i=1 to ivar(5,zr%)+1 :
     rex(i)=re(i+100) : next i
27260 if ziel=2 then for i=1 to ivar(5,zr%)+1 :
     rey(i)=re(i+100) : next i
27270 return
27280 if ziel=1 then for i=1 to an%+1 :
     rex(i)=dy(2,i+100) : next i
27285 if ziel=2 then for i=1 to an%+1 :
     rey(i)=dy(2,i+100) : next i
27290 return
27300 lzr%=zr% : if leseflag%=1 then goto 27380
27310 an%=ivar(5,zr%) : gosub 9200 : rem ********** get
     yreg******************
27350 if ziel=1 then for i=1 to ivar(5,zr%)+1 :
     rex(i)=re(i) : next i
27360 if ziel=2 then for i=1 to ivar(5,zr%)+1 :
     rey(i)=re(i) : next i
27370 return
27380 if ziel=1 then for i=1 to an%+1 : rex(i)=dy(2,i)
     : next i
27385 if ziel=2 then for i=1 to an%+1 : rey(i)=dy(2,i)
     : next i
27390 return
29800 print "Interpreter error" : goto 25990
29900 print "wrong input" : goto 25990


rem ******************input routine*******************
40000 icol%=col%+len(prompt$) : il=len(in$) : if
     il>ileer or icol%>79 then in$=space$(ileer) :
     return
40020 if icol%+ileer>80 then ileer=80-icol%
40030 if il<ileer then in$=in$+space$(ileer-il)
```

```
40040 call curs(col%,row%) : print prompt$;in$;"*" :
     call curs(icol%,row%) : ia=0 : tshare=100
40050 while ia<>13 : call echo(ia) : if ia>17 and ia<27
     or ia=189 then 40190
40060 rem iplot=iplot+1 : if iplot>tshare then iplot=0
     : if pointer%(1)>0 then gosub 4300
40080 wend
40100 h1$=space$(80)+"" : call readli(row%,h1$)
40120 for ii=1 to ileer : if mid$(h1$,icol%+ii,ileer-
     ii+1)=space$(ileer-ii+1) then 40150
40130 next ii : ii=ileer
40150 in$=mid$(h1$,icol%+1,ii) : h1$=""
40190 return
rem ***************Error Recovery ******************
50000 resume 50010
50010 col%=0 : row%=22 : call curs(col%,row%)
50020 print "Interpretererror  "err;" ocurred on line
     ";erl
50200 print "Hit any Key to proceed" : a$=input$(1)
50270 goto 20000

rem **************************************************
rem ********   MICHFIT software :   pl.bas    **********
rem ********   Author   :   Bruno Michel      ***********
rem
*******************************************************
rem
rem  list of variables and pointers :
rem
rem  **************************************************
rem  plot definitions :
rem
rem  beschr ()      annotate strings, title and labels
                    of axes
```

```
rem  tl ()            string lengths
rem  pg ()  pga ()    plot definitions (see below)
rem  pxu#() pxo#()    minimum, maximum, delta x and delta
                      xo
rem  pyl#() pyr#()    minimum, maximum, delta y and delta
                      yo
rem  pgx () pgy ()    sizes of title, labels, digits,
                      symbols and annotate
rem  idx () idy ()    draw data points (integer x and y)
rem  idp ()           pen, symbol or line type of resp.
                      draw point
rem
*******************************************************
rem  pointer%(ii) functions  :
rem
rem 1 :  No. of plots in outputspool       16 :
rem 2 :  idmax (Max. No. of drawpoints)    17 :
rem 3 :  current macroline   18 :  current direct. page
rem 4 :  Interpreter On off         19 :  current menu
rem 5 : output in process  20 :  initialisation pointer
rem 6 : position during string output 21 :  zoom on/off
rem 7 :                            22 :  graphik on off
rem   8 :  current input page         23 :  display lock
rem   9 :                          24 :  pos neg display
rem  10 :                    25 :  loaded character set 1-5
rem  11 :  Input Menu update          26 :  format loaded
rem  12 :  Output menu update
rem  13 :  Content menu update
rem  14 :  Plotsize menu update
rem  15 :
rem
******************************************************
rem  plot definitions : pg(i)
rem
```

```
rem   1 :  X-size of plot in cm
rem   2 :  Y-size of plot in cm
rem   3 :  Distance from physical left rim of paper in
         cm
rem   4 :  Distance from physical lower rim of paper in
         cm
rem   5 :  Speed of plotter pen in cm/sec
rem   6 :  Size of X-and Y-ticks in % of resp. X- and
         Y-dimensions
rem   7 :  Line spacing between annotate lines in block
         mode
rem   8 :  ---
rem   9 :  Position of Plot depending on setting of
         layout (1-6)
rem  10 :  Output spool for plotter
rem  11 :  Pen of Title (0-8) 0: no title
rem  12 :  Pen of Axes (0-8)  0: no axes
rem  13 :  Layout of plot (1-4) along 1, across 2,
         along 4, across 6
rem  14 :  Number of annotate lines (0-15)
rem  15 :  Pen of annotate (0-8)
rem  16 - 30 : orientation of the annotate labels
         1=hoizontal 2=vertical
rem   3=horizontal on top 4=vertical from top to bottom
rem  **************************************************
rem  Plot definitions : pga (i)
rem
rem
rem   8 :  Width of columns in % of the x-dimension of
     the plot
rem
rem
rem
rem  **************************************************
rem  Plot definitions : pgx (i), pgy (i)
rem   1 -  5 :      Sizes of Title, labels of axes,
     digits, symbols and annotate
rem   6 - 10 :  Position of Labels in % relative to the
     frame of the plot
rem        1) title 2) label of lower x-axis 3)label of
     left y-axis
rem        4) label of upper x-axis 5) label of right y-
     axis
rem  11 - 15 :  Position of digits in units of
     digitsize relative to the
rem        intersection of the tick with the frame :
rem        1) -- 2) lower x-axis 3) left y-axis 4) upper
     x-axis
rem        5) right y-axis
rem  16 - 30 :  position of the annotate labels in %
     relative to the plot frame
rem
*******************************************************
rem                    Start of source code
rem
*******************************************************
rem  **************** definitions ********************
30 rem $include : 'comvar'
220 width 255 :
250 drt$="a:b:c:d:e:f:g:h:"
255 scrndr$=mid$(drt$,idrive(1)*2-1,2):
     spooldr$=mid$(drt$,idrive(3)*2-1,2)
260 sysdr$=mid$(drt$,idrive(4)*2-1,2) :
     formdr$=mid$(drt$,idrive(5)*2-1,2)
265 mapgdr$=mid$(drt$,idrive(7)*2-1,2)
300 close#11 : open "LPT1:" for output as#11 len=80
305 field#14,1760 as screen$
310 dim rex(500),rey(500),dislin%(10),texept(20)
```

```
315 tsset=1 : cursor=1 : cw=1 : idmax=0 : idc=1 :
     pga(9)=1 : plotmen%=0 : iannot=0 : back%=0 :
     upper%=1 : touchs=0 : ttaus=taus
320 running%=1 : isym=1 : ileer=5 : insym=1 : tshare=2
     : iplot=1 : aufspulen%=0
325 for i=1 to 20 : texept(i)=0 : next i
326 texept(1)=228 : texept(2)=215 : texept(3)=227 :
     texept(4)=250 : texept(5)=186 : texept(6)=181 :
     texept(7)=208 : texept(8)=187 : texept(9)=240
327 texept(11)=204 : texept(12)=206 : texept(13)=207
rem *********** read systemfile with prompts **********
330 if pointer%(26)=1 then 350
335 close#2 : open "r",#2,name$+".HSY",256 :
     field#2,256 as ex$
340 for i=11 to 20 : get#2,i : il=cvi(mid$(ex$,1,2)) :
     format$(i+68)=mid$(ex$,3,il) : next i
345 if pointer%(25)=0 then i=1 : gosub 26410 else
     close#2 : ex$=""
rem *********** start in different points ***********
350 gosub 19100 : gosub 20000 : if pointer%(21)=0 then
     pointer%(23)=0
360 if pinit<1 or pinit>2 then 1900
370 if autoscale=1 and pointer%(21)=1 and
     pointer%(23)=0 then pointer%(22)=0 : for block%=1
     to 6 : gosub 22030 : next block%
375 if autoscale=1 then gosub 27000
380 if interpolate=1 then for j=1 to 5 : smooth=2 :
     gosub 27500 : gosub 26500 : next j
390 ttaus=taus : if pinit=2 then ime=2 : gosub 7000 :
     if taus>1 and taus<5 then running%=0
400 if pinit=1 then gosub 21000
410 pinit=1 : for i=61 to 72 : ip(i)=1 : next i
rem  **************************************************
rem                      Main program
```

```
rem
*******************************************************
500 while running%
rem 505 if warnz%>0 then gosub 6300
rem 510 if comfile%>0 then a$=macro$(comfile%) :
     comfile%=comfile%+1 : if comfile%>21 then
     comfile%=0
rem 515 if comfile%>0 then gosub 4500
rem 520 if comfile%=0 and interpreter=1 then gosub 4500
540 tshare=2 : ilock=0
550 gosub 5000 : rem ********wait for input *****
555 if running%=0 then goto 790
560 if ime<1 or ime>2 then ime=1
565 if iflag=0 then 700
570 on ime gosub 21400,22400
700 on ime gosub 21025,2000
790 wend
rem *************back to calling pgm*****************
1000 ttaus=taus : if pointer%(21)=1 and pointer%(23)=0
     then pointer%(22)=0 : block%=1 : gosub 22030 :
     print groff$; : call alpha
1005 mid$(beschr$(1),1,len(xbes$))=xbes$ :
     tl(1)=len(xbes$)
1010 mid$(beschr$(2),1,len(ybes$))=ybes$ :
     tl(2)=len(ybes$)
1020 mid$(beschr$(3),1,len(xobes$))=xobes$ :
     tl(3)=len(xobes$)
1030 mid$(beschr$(4),1,len(yrbes$))=yrbes$ :
     tl(4)=len(yrbes$)
1040 mid$(beschr$(5),1,len(ptitel$))=ptitel$ :
     tl(5)=len(ptitel$)
1050 close#1,#2 : promptl$="" : fstrl$="" : fstr2$="" :
     h$="" : h1$="" : h2$="" : in$="" : insav$=""
```

```
1070 formpl$="" : textpl$="" : n$="" : m$="" :
     zeichen$="" : pl$="" : ex$="" : print groff$; :
     print at$;
1075 call alpha : if plotmen%=1 and ip(90)=1 then gosub
     21150
1080 call cls : ime=0
1081 if forts>3 then 1090 else if forts<1 then forts=2
1082 if forts=1 then get#14,1 : ime=1 : iil=1
1083 if forts=2 then get#14,2 : ime=2 : iil=31
1084 if forts=3 then get#14,4+anzeige : ime=3 :
     iil=61+(softk-1)*2
1085 def seg : il=varptr(#14) : call recscr(il) : gosub
     8500
1086 il=forts : softkey$=format$(iil)+format$(iil+1) :
     gosub 8700
1090 ttaus=1 : if pointer%(21)=1 and pointer%(23)=0
     then for block%=1 to 6 : gosub 22030 : next block%
     : print at$; : print groff$;
1091 ptitel$="" : xbes$="" : ybes$="" : xobes$="" :
     yrbes$="" : groff$="" : clearm$="" : setm$="" :
     togglem$=""
1092 softkey$="" : sk$="" : i=fre("") :
     pointer%(19)=ime
1095 pointer%(2)=idmax
1100 pinit=1 : chain pret$
1900 end
2000 return
rem
*****************************************************
rem                  Subroutines  /  procedures
rem *************************************************
rem **************** Input 1 or 2 numbers  ************
3960 ileer=ispa : gosub 40000
```

```
3970 a$="," : ikg=instr(in$,a$) : if ikg=0 then
     h1=val(in$) : h2=0 else h1=val(mid$(in$,1,ikg-1))
     : h2=val(mid$(in$,ikg+1,len(in$)-ikg))
3975 return
rem ***********background output********************
4300 if pointer%(5)<1 then 4320
4305 if eof(13) and aufspulen%=1 then return
4310 if not eof(13) then input#13,io$ else gosub 4400 :
     return
4312 if len(io$)<10 then io$=io$+space$(32)
4315 print#12,io$
4320 return
4350 scol%=0 : srow%=21 : call curs(scol%,srow%) :
     print space$(79) : call curs(scol%,srow%) : print
     "No. of Plots :";pointer%(1);" repeat
     (0/1)";repeat%;" hit space " : a$=input$(1)
4355 call curs(scol%,srow%) : print space$(79) : call
     curs(scol%,srow%) : print "Please enter command
     for plot spool :"
4356 print "e=exit / d=delete / r=repeat / c=clear /
     p=pause / s=start" : a$=input$(1) : on error goto
     6000
4360 if a$="d" then if pointer%(5)=1 then gosub 4400
     else gosub 4402
4365 if a$="r" then repeat%=1
4370 if a$="c" then for j=1 to pointer%(1) : kill
     spooldr$+"PLOT"+chr$(j+64) : next j :
     pointer%(1)=0 : pointer%(5)=0
4375 if a$="p" then close#12 : close#13 : pointer%(5)=0
4380 if a$="s" then if pointer%(1)>0 then pointer%(5)=1
     : gosub 4415
4390 return
rem ***********close open Spool**********************
```

```
4400 print#12,"PU PA 0,0;SP 0;" : close#12 : close#13 :
     if repeat%=1 then repeat%=0 : goto 4420
4402 kill spooldr$+"PLOT"+chr$(1+64) : pointer%(1)=
     pointer%(1)-1 : if pointer%(1)=0 then
     pointer%(5)=0 : goto 4425
4405 for ips=1 to pointer%(1) : name spooldr$+"PLOT"+
     chr$(ips+1+64) as spooldr$+"PLOT"+chr$(ips+64) :
     next ips
4415 open "o",#12,"PLT" : width #12,20 : open
     "i",#13,spooldr$+"PLOT"+chr$(1+64)
4420 input#13,io$ : if io$="STOP" then scol%=0 :
     srow%=22 : call curs(scol%,srow%) : print "Hit
     Enter when Plotter ready (s=stop)" : a$=input$(1)
     : if a$="s" then pointer%(5)=0 : close#12 :
     close#13
4425 return
rem **** Keyboard Touchscreen Mouse input  ***********
5000 if touchs=0 and ilock=0 then print chr$(27);"-
     z2N"; : touchs=1 : rem **touchsense on ********
5005 iflag=0 : iplot=iplot+1 : if iplot>tshare then
     iplot=0 : if pointer%(1)>0 then gosub 4300
5010 a1$=inkey$ : if len(a1$)<1 then a$=" " : b=0 : a=0
     : return else a$=mid$(a1$,1,1) : a=asc(a$)-17 :
     b=asc(a$)-89 : iflag=1
5015 print chr$(27); "-z0N"; : touchs=0 : rem
     ******touchsense off **********************
5020 tfeld=0 : tarray=0 : tsoft=0 : tcol=0 : trow=0 :
     icol=0 : irow=0 : rem if eing%=1 then eing%=0 :
     goto 5520
5025 if asc(a$)>17 and asc(a$)<27 then tsoft=asc(a$)-17
     : return
5027 if ilock=1 then return
5220 if a$="*" then forts=1 : running%=0 : return
5230 if a$="/" then forts=2 : running%=0 : return
```

```
5240 if a$="+" then forts=3 : running%=0 : return
5250 if a$="-" then plotmen%=0 : gosub 21000
5300 return : rem *************** back to main program
     *********************
rem **********error recovery***************************
6000 resume 6010
6010 gosub 22390 : gosub 22350 : call curs(col%,row%)
6020 if erl=25800 then print "plotter not responding";
     : goto 6090
6030 if erl>24000 and erl<25000 then print "printer not
     responding"; : goto 6090
rem 6040 if erl=
6080 print "error ";ern;" ocurred on line ";erl;
6090 print " (hit any key to proceed)" : a$=input$(1) :
goto 500
rem ************find output ***************************
7000 ttaus=taus : if taus=2 then aus$=name$ : gosub
25000 : return
7010 if taus=3 then aus$=name$ : gosub 24000 : return
7020 if taus=4 then pointer%(1)=pointer%(1)+1 :
     n$=spooldr$+"PLOT"+chr$(pointer%(1)+64) : on error
     goto 21900 : ttaus=2
7030 if taus=4 then close #1 : open "o",#1,n$ :
     print#1,"NORMAL" : aufspulen%=1 : gosub 25010 :
     aufspulen%=0 : return
7040 if taus=5 then forts=8 : running%=0 : return
7050 if taus=1 then ime=2 : tsset=1 : a$=" " : a=0 :
     call alphaoff : print chr$(27)+"*dS"; : gosub
     22000 : call graph : return
     7090 return
rem ************** setup Touchscreen ******************
8500 call offtouch
8540 for i=1 to 25 : i1=1+(i-1)*2+(ime-1)*100 : i2=i+65
8550  if icon(i1)=icon(i1+1) then 8600
```

```
8560  col%=icon(i1)/100 : row%=icon(i1)-col%*100
8570  colinc%=icon(i1+1)/100 : rowinc%=icon(i1+1)-
     colinc%*100-row% : colinc%=colinc%-col%
8580  resp$=chr$(i2) : call fntouch(row%,col%,
     rowinc%,colinc%,resp$)
8590 next i
8600 ipoint=(ime-1)*100+51 : i2=89
8610 i=ipoint : if icon(i)=0 then 8695
8620 if icon(i)>9899 then rowoff%=icon(i)-9900 :
     anzz%=icon(i+1)/100 : anzs%=icon(i+1)-anzz%*100
8630 for i=1 to anzs% : i1=ipoint+i*2 :
     col%=icon(i1)/100 : row%=icon(i1)-col%*100
8640  colinc%=icon(i1+1)/100 : rowinc%=icon(i1+1)-
     colinc%*100-row% : colinc%=colinc%-col%
8650  for j=1 to anzz% : i2=i2+1 : rov%=row%+(j-
     1)*rowoff%
8670   resp$=chr$(i2) : call fntouch(rov%,col%,
     rowinc%,colinc%,resp$)
8680  next j
8690 next i : ipoint=ipoint+anzs%*2+2 : goto 8610
8695 return
rem
************Keylabel***********************************
8700 if len(softkey$)<160 then softkey$=softkey$
     +space$(161-len(softkey$))
8705 for i=1 to 8 : m$=str$(i) : sk$=chr$(27)+"&f0a
     "+mid$(m$,2,1)+"k16d1L"+mid$(softkey$,i*18-
     15,16)+chr$(17+i) : print sk$; : next i
8710 print chr$(27); "&jB"; : return
rem
***************disli***********************************
8900 col%=0 : call curs(col%,row%) : print h1$ : return
     rem 8900 enh$=chr$(255) : call disli(row%,   h1$,e
     nh$) : return
```

```
19100 rem *********demultiplex Strings*****************
19110 xbes$=mid$(beschr$(1),1,tl(1)) :  ybes$=mid$(besc
     hr$(2),1,tl(2))
19120 xobes$=mid$(beschr$(3),1,tl(3)) : yrbes$=mid$
     (beschr$(4),1,tl(4))
19130 ptitel$=mid$(beschr$(5),1,tl(5)) :
     idmax=pointer%(2)
19180 return
rem  ******** definitions for plotmodule **************
20000 rem
20220 format=1 : cursor=1 : cw=4 : cursg=1 : x%=300 :
     y%=200
20230 dislin%(1)=1 : dislin%(2)=7 : dislin%(3)=6 :
     dislin%(4)=5 : dislin%(5)=8
20240 dislin%(6)=4 : dislin%(7)=10 : dislin%(8)=9 :
     dislin%(9)=11 : dislin%(10)=1
20260 togglem$=chr$(27)+"*m3A" : gt$=chr$(27)+"*dS" :
     at$=chr$(27)+"*dT"
20270 prompt1$="Geben Sie : 1=Ticks 2=Zahlen 4=Beschr."
     : Zeichen$="x*+-=OO%/#@"
20300 groff$=chr$(27)+"*dD"+chr$(27)+"*dT"
20320 def fnplu$(x,y)=chr$(27)+"*pa"+
     str$(int(pxmin+scxf#*(x-scadx#)))+","+str$(int
     (pymin+scyf#*(y-scady#)))+"Z"
20330 def fnpld$(x,y)=chr$(27)+"*pb"+str$(int
     (pxmin+scxf#*(x-scadx#)))+","+str$(int(pymin+scyf#
     *(y-scady#)))+"Z"
20340 def fnpmu$(x,y)="pu;pa"+str$(int(pxmin+(x-
     scadx#)*scxf#))+","+str$(int(pymin+(y-
     scady#)*scyf#))+";"
20350 def fnpm$(x,y)="pa;pd"+str$(int(pxmin+(x-
     scadx#)*scxf#))+","+str$(int(pymin+(y-
     scady#)*scyf#))+";"
```

```
20360 def fnplu2$(x,y)=chr$(27)+"*pa"+str$(int(pzmax-
     scyf#*(y-scady#)))+","+str$(int(pymin+scxf#*(x-
     scadx#)))+"Z"
20370 def fnpld2$(x,y)=chr$(27)+"*pb"+str$(int(pzmax-
     scyf#*(y-scady#)))+","+str$(int(pymin+scxf#*(x-
     scadx#)))+"Z"
20380 def fnpmu2$(x,y)="pu;pa"+str$(int(pzmax-(y-
     scady#)*scyf#))+","+str$(int(pymin+(x-
     scadx#)*scxf#))+";"
20390 def fnpm2$(x,y)="pa;pd"+str$(int(pzmax-(y-
     scady#)*scyf#))+","+str$(int(pymin+(x-
     scadx#)*scxf#))+";"
20400 def fnxw(x)=int(pxmin+scxf#*(x-scadx#)) : def
     fnyw(y)=int(pymin+scyf#*(y-scady#))
20420 def fnscx(scxmin#,scxmax#)=(pxmax-
     pxmin)/(scxmax#-scxmin#)
20430 def fnscy(scymin#,scymax#)=(pymax-
     pymin)/(scymax#-scymin#)
20440 def fncurs$(x,y)=chr$(27)+"*d"+str$
     (int(pxmin+scxf#*(x-scadx#)))+","+str$(int
     (pymin+scyf#*(y-scady#)))+"O"
20450 def fnw$(x,y)=str$(int(pxmin+(x-scadx#)*scxf#))
     +","+str$(int(pymin+(y-scady#)*scyf#))
20460 def fnv$(x,y)=str$(int(pzmax-(y-scady#)*scyf
     #))+","+str$(int(pymin+(x-scadx#)*scxf#))
20495 gosub 20500 : return
20500 if pointer%(24)=0 then clearm$=chr$(27)+"*m1A" :
     setm$=chr$(27)+"*m2A" else clearm$=chr$(27)+"*m2A"
     : setm$=chr$(27)+"*m1A"
20510 return
rem *************** Plotsize menu ********************
rem
21000 ime=1 : tshare=2 : call graphoff : call alpha
```

```
21005 softkey$="1=PositiveNegative2= Scale   Autom. 3=
     Next   Position4=Execute Ausgabe 5=AnnotateEingabe
     6=Ausgabe abspulen7= Upper   lower  8= Rolle
     Beschr. "
21010 i1=1 : gosub 8700 : if plotmen%<>1 then gosub
21020
21015 return
21020 on error goto 0 : call cls : call offtouch : def
     seg : get#14,4 : i1=varptr(#14) : call recscr(i1)
     : gosub 30000
21022 gosub 30000 : ip(90)=0 : pointer%(14)=0 :
     plotmen%=1 : for i=73 to 86 : ip(i)=1 : next i :
     col%=0 : row%=0 : call curs(col%,row%) : call
     alpha : return
21025 if pointer%(14)>30 then return
21030 pointer%(14)=pointer%(14)+1 : if ip(pointer%(14)
     +60)=0 then return
21031 if pointer%(14)=1 then print at$;
21032 if upper%=1 then if pointer%(14)>12 and
     pointer%(14)<28 then return
21033 if upper%=0 then if pointer%(14)<13 or
     pointer%(14)=30 then return
21035 on error goto 0 : ip(pointer%(14)+60)=0 : if
     pointer%(14)>20 then 21050
21040 on pointer%(14) gosub 21200,21210,21220,21230,
     21240,21250,21260,21270,21280,21290,21300,21310,21
     320,21330,21340,21350,21100,21100,21100,21100
rem                           1      2      3      4      5
     6      7      8      9     10     11     12     13     14
     15     16     17     18     19     20
21045 return
21050 ii=pointer%(14)-20 : on ii gosub
21360,21360,21360,21360,21370,21380,21100,21190,21190,
     21150
```

```
rem                                          1     2     3
    4     5     6     7     8     9     10
21055 return
21100 return
21150 def seg : i1=varptr(#14) : call stoscr(i1) :
    put#14,4 : return
21190 col%=0 : row%=21 : call curs(col%,row%) : print
    space$(159) : return
rem *************Upper part****************************
21200 col%=20 : row%=1 : call curs(col%,row%) : print
    ptitel$;space$(80) : return
21210 col%=20 : row%=3 : call curs(col%,row%) : print "
    ";mid$(format$(83),taus*7-6,7);" " : return
21220 col%=68 : row%=3 : call curs(col%,row%) : print "
    ";pg(11);"  ";pg(12);"  " : return
21230 col%=20 : row%=5 : call curs(col%,row%) : print
    mid$(format$(84),pg(13)*16-15,12);pg(9) : return
21240 col%=68 : row%=5 : call curs(col%,row%) : print "
    ";pg(14);"  ";pg(15);"  " : return
21250 col%=11 : row%=8 : call curs(col%,row%) : h1$="
    " : for i=1 to 7 : h2$=str$(pg(i))+space$(9) :
    h1$=h1$+mid$(h2$,1,8) : next i : h1$=h1$+str$
    (pga(8)) : print h1$ : return
21260 col%=8 : row%=10 : call curs(col%,row%) : print
    pxu#(5) : for i=1 to 4 : h1=csng(pxu#(i)) : gosub
    21315 : next i
21265 col%=44 : call curs(col%,row%) : print "
    ";xbes$;space$(79) : return
21270 col%=8 : row%=12 : call curs(col%,row%) : print
    pyl#(5) : for i=1 to 4 : h1=csng(pyl#(i)) : gosub
    21315 : next i
21275 col%=44 : call curs(col%,row%) : print "
    ";ybes$;space$(79) : return
```

```
21280 col%=8 : row%=14 : call curs(col%,row%) : print
     pxo#(5) : for i=1 to 4 : h1=csng(pxo#(i)) : gosub
     21315 : next i
21285 col%=44 : call curs(col%,row%) : print "
     ";xobes$;space$(79) : return
21290 col%=8 : row%=16 : call curs(col%,row%) : print
     pyr#(5) : for i=1 to 4 : h1=csng(pyr#(i)) : gosub
     21315 : next i
21295 col%=44 : call curs(col%,row%) : print "
     ";yrbes$;space$(50-len(yrbes$)) : return
21300 row%=18 : for i=1 to 5 : col%=i*8+3 : call
     curs(col%,row%) : print "  ";pgx(i);"  " : next i
     : return
21310 row%=20 : for i=1 to 5 : col%=i*8+3 : call
     curs(col%,row%) : print "  ";pgy(i);"  " : next i
     : return
21315 col%=i*8+3 : call curs(col%,row%) : h1$=str$(h1)
     : i2=len(h1$) : if i2<7 then print "
     ";h1$;space$(7-i2) : else print using
     "+#.##^^^^";h1
21317 return
rem ***************Lower Part***************************
21320 row%=26 : h1$="x-coord. text     " : for i=6 to
     10 : h2$=str$(pgx(i))+space$(10) :
     h1$=h1$+mid$(h2$,1,10) : next i : gosub 8900 :
     return
21330 row%=28 : h1$="y-coord. text     " : for i=6 to
     10 : h2$=str$(pgy(i))+space$(10) : h1$=h1$+
     mid$(h2$,1,10) : next i : gosub 8900 : return
21340 row%=30 : h1$="x-Pos. digits     " : for i=11 to
     15 : h2$=str$(pgx(i))+space$(10) :
     h1$=h1$+mid$(h2$,1,10) : next i : gosub 8900 :
     return
```

```
21350 row%=32 : h1$="y-pos. digits     " : for i=11 to
     15 : h2$=str$(pgy(i))+space$(10) :
     h1$=h1$+mid$(h2$,1,10) : next i : gosub 8900 :
     return
rem *******************annotate *********************
21360 i=pointer%(14)-20 : h1$=space$(50) :
     h2$=space$(50) : col%=0 : i1=i+roll% : if i1>15
     then 21368
21362 il=tl(i1+5) : row%=(i-1)*2+35 : if il>50 then
     mid$(h1$,1,50)=mid$(beschr$(i1+5),1,50) :
     mid$(h2$,1,il-51)=mid$(beschr$(i1+5),51,il-51)
21364  if il<51 and il>0 then mid$(h1$,1,il)=
     mid$(beschr$(i1+5),1,il)
21368 call curs(col%,row%) : print using
     format$(79);i1,pg(i1+15),pgx(i1+15),pgy(i1+15),h1$
     : row%=row%+1 : call curs(col%,row%) : print using
     format$(80);h2$ : return
21370 col%=0 : row%=25 : call curs(col%,row%) : print
     format$(81) : return
21380 col%=0 : row%=34 : call curs(col%,row%) : print
     format$(82) : return
rem ******* subroutines for plotsize menu commands ****
21400 col%=0 : row%=22 : call curs(col%,row%) : print
     at$; : tshare=2 : ic=asc(a$) : tarray=0 :
     pointer%(22)=0 : if ic>89 and ic<98 then tarray=1
21405 if ic>97 and ic<130 then tarray=2
21410 if ic>129 and ic<146 then tarray=3
21415 if ic>145 and ic<170 then tarray=4
21420 if ic>169 and ic<186 then tarray=5
21425 if a>0 and a<9 then 21700
21427 if ic>185 or ic<65 then return
rem ************* actions of single fields ***********
21430 ip(90)=1 : ip(89)=1 : pointer%(14)=0 : on tarray
     goto 21600,21600,21600,21650,21680
```

```
21440 on error goto 21900 : h1$="TQNAXYVWM" :
     ii=instr(h1$,a$)
21445 if ii>4 and ii<9 then block%=2 : gosub 22030 :
     block%=5 : gosub 22030 : print at$; : print
     prompt1$;
21450 on ii gosub
21460,21470,21480,21490,21500,21510,21520,21530,21540,
     21550
21455 return
21460 col%=20 : row%=1 : prompt$="?" : in$=ptitel$ :
     ileer=58 : gosub 40000 : insav$=in$ : if
     len(insav$)>80 then goto 21450 else ip(61)=1
21465 if pointer%(21)=1 then block%=2 : gosub 22030 :
     print at$;
21467 ptitel$=insav$ : return
21470 col%=69 : row%=3 : call curs(col%,row%) :
     prompt$="?" : in$="_,_" : ispa=3 : gosub 3960 :
     sav1=h1 : sav2=h2 : ip(63)=1
21475 if pointer%(21)=1 then block%=1 : gosub 22030 :
     block%=2 : gosub 22030 : print at$;
21477 pg(11)=sav1 : pg(12)=sav2 : return
21480 col%=69 : row%=5 : call curs(col%,row%) :
     prompt$="?" : in$="__,_" : ispa=4 : gosub 3960 :
     sav1=h1 : sav2=h2
21485 if pointer%(21)=1 then block%=3 : gosub 22030 :
     print at$;
21487 pg(14)=sav1 : pg(15)=sav2 : ip(65)=1 : return
21490 taus=taus+1 : if taus>5 then taus=1
21495 ip(62)=1 : return
21500 gosub 21580 : pxu#(5)=sav# : ip(67)=1 : return
21510 gosub 21580 : pyl#(5)=sav# : ip(68)=1 : return
21520 gosub 21580 : pxo#(5)=sav# : ip(69)=1 : return
21530 gosub 21580 : pyr#(5)=sav# : ip(70)=1 : return
21540 pg(13)=pg(13)+1 : if pg(13)>4 then pg(13)=1
```

```
21545 gosub 29800 : pg(9)=1 : gosub 29500 : ip(64)=1 :
     ip(66)=1 : for i=71 to 76 : ip(i)=1 : next i : if
     pointer%(21)=1 then call gclear
21547 pointer%(23)=0 : for i=91 to 99 : ip(i)=0 : next
     i : return
21550 return
21580 input sav#
21585 if pointer%(21)=1 then block%=1 : gosub 22030 :
     block%=2 : gosub 22030 : print at$;
21587 return
rem *************** actions of field arrays ***********
21600 tcol=b mod 8 : if tcol=0 then tcol=8
21605 trow=int((b-tcol)/8+1) : tcolsav=tcol :
     trowsav=trow
21610 prompt$="?" : in$=" " : ileer=5 : if trow=1 or
     tcol<5 or trow>5 then row%=6+trow*2 :
     col%=tcol*8+4 : gosub 40000 : insav$=in$
21620 if pointer%(21)=1 then h1$="please wait" : gosub
21950 : for block%=1 to 9 : gosub 22030 : next block% :
     print at$; else for i=91 to 99 : ip(i)=0 : next i
21630 trow=trowsav : tcol=tcolsav : col%=0 : row%=22 :
     ileer=75 : in$=insav$
21640 if trow=1 then ip(66)=1 : h1=val(in$) : if tcol=8
     then pga(8)=h1 else pg(tcol)=h1
21642 if trow=2 then ip(67)=1 : if tcol=5 then
     in$=xbes$ : gosub 40000 : xbes$=in$ else
     pxu#(tcol)=val(in$)
21643 if trow=3 then ip(68)=1 : if tcol=5 then
     in$=ybes$ : gosub 40000 : ybes$=in$ else
     pyl#(tcol)=val(in$)
21644 if trow=4 then ip(69)=1 : if tcol=5 then
     in$=xobes$ : gosub 40000 : xobes$=in$ else
     pxo#(tcol)=val(in$)
```

```
21645 if trow=5 then ip(70)=1 : if tcol=5 then
      in$=yrbes$ : gosub 40000 : yrbes$=in$ else
      pyr#(tcol)=val(in$)
21646 if trow=6 then ip(71)=1 : pgx(tcol)=val(in$)
21647 if trow=7 then ip(72)=1 : pgy(tcol)=val(in$)
21648 return
21650 c=asc(a$)-145 : tcol=c mod 6 : if tcol=0 then
      tcol=6
21660 trow=int((c-tcol)/6+1) : row%=24+trow*2 :
      col%=tcol*10+9 : prompt$="?" : in$=" " : ileer=5 :
      gosub 40000
21661 trowsav=trow : tcolsav=tcol : insav$=in$
21662 if pointer%(21)=1 then for block%=1 to 5 : gosub
22030 : next block% : print at$; else for i=92 to 95 :
      ip(i)=0 : next i
21663 trow=trowsav : tcol=tcolsav : in$=insav$
21665 if trow=1 then pgx(tcol+5)=val(in$) : ip(73)=1
21670 if trow=2 then pgy(tcol+5)=val(in$) : ip(74)=1
21675 if trow=3 then pgx(tcol+10)=val(in$) : ip(75)=1
21677 if trow=4 then pgy(tcol+10)=val(in$) : ip(76)=1
21678 return
21680 c=asc(a$)-169 : if c>20 then return
21682 prompt$="?" : tcol=c mod 4 : if tcol=0 then
      tcol=4
21683 trow=int((c-tcol)/4+1) : col%=(tcol-1)*8+1 :
      row%=(trow-1)*2+35 : if trow>5 then return
21684 i=trow+roll% : if tcol<4 then in$="_" : ileer=5 :
      gosub 40000
21685 tcolsav=tcol : trowsav=i : insav$=in$ : block%=3
      : gosub 22030 : print at$;
21686 i=trowsav : tcol=tcolsav : in$=insav$ : if tcol=1
      then pg(i+15)=val(in$)
21688 if tcol=2 then pgx(i+15)=val(in$)
21690 if tcol=3 then pgy(i+15)=val(in$)
```

```
21692 if tcol=4 then col%=0 : row%=22 : ileer=75 :
     in$=mid$(beschr$(i+5),1,tl(i+5)) : gosub 40000 :
     il=len(in$) : if il>80 then return
21693 if tcol=4 then call curs(col%,row%) : print
     space$(79) : tl(i+5)=il : mid$(beschr$(i+5),1,il)
     =in$
21694 for i=81 to 84 : ip(i)=1 : next i
21696 return
rem *** routines performing softkey functions *********
21700 ip(90)=1 : ip(89)=1 : on a gosub
21720,21740,21760,21780,21800,21820,21840,21860
21710 pointer%(14)=0 : return
21720 if pointer%(24)=0 then pointer%(24)=1 else
     pointer%(24)=0
21725 gosub 20500 : call gclear : for i=91 to 99 :
     ip(i)=0 : next i : pointer%(23)=0 : print at$; :
     print "done" : return
21740 col%=0 : row%=22 : call curs(col%,row%) : gosub
27000 : for i=67 to 70 : ip(i)=1 : next i : return :
     rem autoscale
21760 if pg(13)<1 or pg(13)>4 then pg(13)=1
21762 on pg(13) goto 21764,21766,21768,21770
21764 goto 21775
21766 if pg(9)=1 then pg(9)=2 : goto 21775 else pg(9)=1
     : goto 21775
21768 pg(9)=pg(9)+1 : if pg(9)>4 then pg(9)=1 : goto
21775 else goto 21775
21770 pg(9)=pg(9)+1 : if pg(9)>6 then pg(9)=1 : goto
21775 else goto 21775
21775 ttaus=taus : if pointer%(21)=1 then hl$="keep
     plot (y/n)" : gosub 21950 : a$=input$(1) : if
     a$="y" then pointer%(22)=1
```

```
21776 if pointer%(21)=1 then call alphaoff : for
     block%=1 to 9 : gosub 22030 : next block% : call
     graphoff : call alpha
21778 gosub 29500 : ip(64)=1 : ip(66)=1 :
     pointer%(23)=0 : for i=91 to 99 : ip(i)=0 : next i
     : return
21780 gosub 7000 : return
21800 input "Please enter direction of block (1 or
     2)";sav1
21802 call curs(col%,row%) : input "Please enter
     coordinates of upper left corner of the
     block";sav3,sav4
21804 call curs(col%,row%) : input "Plese enter first
     line and line spacing eg. 1,5 ";sav5,sav2
21805 block%=3 : gosub 22030 : print at$; : pg(7)=sav2
     : h1=sav1 : h3=sav3 : h4=sav4 : h5=sav5
21806 for i=h5 to 15 : pg(i+15)=h1 : if h1=1 then
     pgx(15+i)=h3 : pgy(15+i)=h4-pg(7)*(i-h5) else
     pgy(15+i)=h4 : pgx(15+i)=h3-pg(7)*(i-h5)
21810 call curs(col%,row%) : print space$(79) :
     prompt$="?" : in$="" : ileer=75 : gosub 40000 :
     il=len(in$)
21812 if il=0 then 21818 else tl(i+5)=il :
     mid$(beschr$(i+5),1,il)=in$ : pg(14)=i
21814 next i
21818 for i=81 to 84 : ip(i)=1 : next i : return
21820 gosub 4350 : return
21840 pointer%(14)=0 : if upper%=1 then row%=44 :
     upper%=0 else upper%=1 : row%=0
21845 call curs(col%,row%) : return
21860 roll%=roll%+3 : if roll%>12 then roll%=0
21865 ip(81)=1 : ip(82)=1 : ip(83)=1 : ip(84)=1 :
     return
```

```
21900 col%=0 : row%=22 : call curs(col%,row%) : print
      groff$ : print "Error ";err;"  occurred on line :
      ";erl
21910 for k=1 to 4000 : i=k : next k : resume 500
rem **************message kz************************
21950 icol=0 : irow=22 : call curs(icol,irow) : print
      h1$; : if len(h1$)<79 then  print space$(79-
      len(h1$))
21960 call curs(col%,row%) : return
rem ******************** Display menu ****************
rem
22000 ime=2 : tsset=1 : call alphaoff : print gt$;
22010 print togglem$ : pointer%(22)=1
22020 for block%=1 to 10 : gosub 22030 : next block% :
      gosub 22100
22025 if pointer%(23)=1 then gosub 22390 : gosub 22350
      : print "display locked "
22027 return
22030 if pointer%(21)=1 and pointer%(23)=1 then return
22032 if ip(block%+90)=pointer%(22) then return else
      ip(block%+90)=pointer%(22)
22035 on block% gosub
23000,23100,23090,23700,23050,23250,22105,22105,22105,2
      2105
22040 return
rem ************* Define softkey labels ***************
22100 on tsset gosub
22110,22120,22130,22135,22140,22145
22105 return
22110 softkey$="1= Unlock Display 2=  Draw
      3=Select   Symbol 4=Plotsize Menu   5=Return
      Programm6=Annotate  Plot  7=Hardcopy        8=
      more   softkeys"
```

```
22115 i1=1 : gosub 22380 : gosub 22150 : gosub 8700 :
     return
22120 softkey$="1=Smooth  Display 2=Interpol Linie
     3=Replace  Symbol 4= Replace  Pen   5= Lock
     Graph  6= Clear   Screen 7= Zoom    on/off 8=first
     softkeys"
22125 i1=2 : gosub 22380 : gosub 22150 : gosub 8700 :
     return
22130 softkey$="1= Input   title  2= Input  X-axis  3=
     Input  Y-axis  4=                5= redraw
     6=Annotate   off  7=Input   text    8=Erase
     Annotate"
22132 i1=3 : gosub 22380 : gosub 22150 : gosub 8700 :
     return
22135 softkey$="1= Enter  Penup   2= Enter  Pendown 3=
     Move    Dcurs  4=Delete  Drawpo. 5= redraw
     6= Enter   Symbol 7=Insert  point   8=  Exit     "
22137 i1=4 : gosub 22380 : gosub 22150 : gosub 8700 :
     return
22140 softkey$="1= Exit          2=  Plot           3=
     Print  table   4=Graphics print  5= File
     6= Spoole          7=            8= Exit         "
22142 i1=5 : gosub 22380 : gosub 22150 : gosub 8700 :
     return
22145 softkey$=space$(144) : return
22150 h1$=" " : for i=1 to 8 : h1$=h1$+mid$(softkey$,
     (i-1)*18+3,8)+" " : if i=4 then h1$=h1$+" "
22151 next i
22152 print chr$(27)+"*m1M" : print chr$(27)+"*m1N" :
     print clearm$;
22154 isx=0 : isy=12 : call plotu(isx,isy) : print h1$
22160 h1$=" " : for i=1 to 8 : h1$=h1$+mid$(softkey$,
     (i-1)*18+11,8)+" " : if i=4 then h1$=h1$+" "
```

```
22161 next i : isx=0 : isy=2 : call plotu(isx,isy) :
     print h1$ : print togglem$;
22166 return
rem **************** implement commandline ************
22300 print chr$(27)+"*m1M" : print chr$(27)+"*m1N"
22305 xx%=0 : yy%=27 : call plotu(xx%,yy%) : print h1$
22310 return
22350 print chr$(27)+"*m1M" : print chr$(27)+"*m1N"
22355 xx%=0 : yy%=27 : call plotu(xx%,yy%)
22360 return
22380 print setm$; : print chr$(27)+"*m1G"; : print
     chr$(27)+"*m 0 0 511 25 E";
22385 print togglem$; : return
22390 print clearm$; : print chr$(27)+"*m1G"; : print
     chr$(27)+"*m 0 25 511 40 E"; : print togglem$;
22395 gosub 22350 : return
rem *********** Softkey sets for display menu ******
22400 ic=1 : tshare=4
22410 on tsset gosub
22500,22900,22800,22700,22600,22650
22420 return
rem ******* Root softkeys **************************
22500 gosub 28100 : if a<0 or a>9 then return
22502 on a gosub
22510,22520,22530,22540,22550,22560,22570,22580
22505 return
22510 gosub 22390 : pointer%(23)=0 : for i=91 to 99 :
     ip(i)=0 : next i : gosub 22000 : return
22520 tsset=4 : gosub 22100 : return
22530 tsset=6 : gosub 22675 : return
22540 gosub 21000 : return
22550 print groff$; : forts=2 : running%=0 : return
22560 tsset=3 : gosub 22100 : gosub 22390 : return
```

```
22570 gosub 22390 : hl$="Please select output device" :
    gosub 22300
22575 tsset=5 : gosub 22100 : return
22580 tsset=2 : gosub 22100 : return
rem *********Hardcopy Softkeys**********************
22600 ic=1 : gosub 28100 : if a<0 or a>9 then return
22602 on a gosub
22610,22615,22620,22625,22630,22635,22640,22645
22605 return
22610 tsset=1 : gosub 22100 : return
22615 ttaus=2 : gosub 22390 : hl$="a=stop  s=skip" :
    gosub 22300 : gosub 25000 : gosub 22390 :
    pointer%(23)=1 : return
22620 aus$=name$ : gosub 24000 : gosub 22390 : return
22625 gosub 22390 : print chr$(27)+"&p7s5dF"; :
    a$=input$(1) : print chr$(27)+"&p3D"; : return
22630 pointer%(23)=1 : taus=5 : ttaus=5 : forts=8 :
    running%=0 : print at$; : call graphoff : call
    alpha : return
22635 gosub 22390 : hl$="n=normal  h=set brake point" :
    gosub 22300 : a$=input$(1) : gosub 22390 :
    hl$="Spoolfile"+str$(pointer%(1)+1)+"wird
    gespeichert" : gosub 22300
22636 n$=spooldr$+"PLOT"+chr$(pointer%(1)+1+64) : on
    error goto 21900 : sav1=pxmin : sav2=pxmax :
    sav3=pymin : sav4=pymax : ttaus=2 : if a$="e" then
    gosub 22390 : return
22637 close #1 : open "o",#1,n$ : aufspulen%=1 : if
    a$="h" then print#1,"STOP" else print#1,"NORMAL"
22638 gosub 25010 : gosub 22390 : pxmin=sav1 :
    pxmax=sav2 : pymin=sav3 : pymax=sav4 : ttaus=taus
22639 pointer%(1)=pointer%(1)+1 : aufspulen%=0 :
    pointer%(23)=1 : return
22640 return
```

```
22645 tsset=1 : gosub 22100 : return
rem **********Select Symbol*****************************
22650 ic=1 : if cursor<10 then cursg=cursor : rem X-Y
     cursor
22655 cursor=11 : gosub 28200
22656 if a>0 and a<9 then tsset=1 : gosub 22695 : gosub
22100 : return
22657 if a$=" " or a$="s" or a$="S" then 22660
22659 return
22660 for j=1 to 49 : tcol=j mod 10 : if tcol=0 then
     tcol=10
22662 trow=(j-tcol)/10 : yy%=45-trow*10 : xx%=tcol*50-
     25
22665 if x%>xx%-40 and x%<xx%+40 and y%>yy%-5 and
     y%<yy%+5 then insym=j : goto 22672
22670 next j : return
22672 isx=350 : isy=55 : call plotu(isx,isy) : print
     "New symbol ";str$(insym) : return
22675 gosub 22695 : isx=0 : isy=55 : call
     plotu(isx,isy) : print "Select with cursor (s
     =select exit:any Softkey)"
22677 if pointer%(25)<>1 then i=1 : gosub 26400
22680 for j=1 to 49 : tcol=j mod 10 : if tcol=0 then
     tcol=10
22682 trow=(j-tcol)/10 : yy%=45-trow*10 : xx%=tcol*50-
     25 : call plotu(xx%,yy%)
22684   if j<11 then print chr$(27)+sym$(j); : goto
     22690
22685   if j>10 and j<21 then l%=dislin%(j-10) : call
     linetype(l%) : xa%=xx%-20 : ya%=yy%-4 : call
     plotu(xa%,ya%) : xa%=xx%+20 : ya%=yy%+4 : call
     plotd(xa%,ya%) : call penup : goto 22690
22686   if j>30 and j<41 then print mid$(Zeichen$,j-
     29,1) : goto 22690
```

```
22688   a$=str$(j) : mid$(a$,1,1)="s" : print a$
22690 next j : return
22695 print clearm$; : print chr$(27)+"*m1G"; : print
     chr$(27)+"*m 0 0 511 65 E"; : print togglem$;
22697 return
rem *********Draw Softkeys ***************************
22700 ic=1 : if cursor<10 then cursg=cursor : rem X-Y
     cursor
22702 cursor=11 : gosub 28200 : if a<0 or a>9 then
     return
22705 on a gosub
22710,22720,22730,22740,22750,22760,22770,22780
22707 return
22710 gosub 23700 : gosub 22785 : idp(idmax)=0 : gosub
     23700
22715 return
22720 gosub 23700 : gosub 22785 : idp(idmax)=1 : gosub
     23700
22725 return
22730 idc=idc+1 : if idc>idmax then idc=1
22735 xh%=idx(idc) : yh%=idy(idc) : gosub 22795 : call
     mgcurs(xh%,yh%) : return
22740 gosub 23700 : if idmax<1 then return
22742 for i=idc to idmax-1 : idx(i)=idx(i+1) :
     idy(i)=idy(i+1) : idp(i)=idp(i+1) : next i
22744 idmax=idmax-1 : idp(idmax+1)=0 : gosub 23700 :
     xh%=idx(idc) : yh%=idy(idc) : gosub 22795 : call
     mgcurs(xh%,yh%) : return
22750 for i=91 to 99 : ip(i)=0 : next i : if
     pointer%(21)=0 then gosub 22000
22755 return
22760 gosub 23700 : gosub 22785 : idp(idmax)=100+insym
     : gosub 23700
22770 return
```

```
22780 cursor=cursg : tsset=1 : gosub 22100 : return
22785 gosub 22790 : idmax=idmax+1 : idx(idmax)=xh% :
     idy(idmax)=yh% : return
22790 xh%=(x%-pxmin)/(pxmax-pxmin)*10000 : yh%=(y%-
     pymin)/(pymax-pymin)*10000 : return
22795 xh%=pxmin+(xh%/10000)*(pxmax-pxmin) :
     yh%=pymin+(yh%/10000)*(pymax-pymin) : return
rem **********Annotate Softkeys************************
22800 ic=1 : if cursor<10 then cursg=cursor : rem X-Y
     cursor
22805 cursor=11 : gosub 28200 : if a<0 or a>9 then
     return
22807 on a gosub
22810,22820,22830,22840,22850,22860,22870,22880
22809 return
22810 gosub 22890 : input "titel";ptitel$ : gosub 22895
     : ip(61)=1 : return
22820 gosub 22890 : input "X-axis";xbes$ : pxu#(5)=7 :
     gosub 22895 : ip(67)=1 : return
22830 gosub 22890 : input "Y-axis";ybes$ : pyl#(5)=7 :
     gosub 22895 : ip(68)=1 : return
22840 return
22850 return
22860 cursor=cursg : tsset=1 : gosub 22100 : return
22870 gosub 22390 : gosub 23090 : il=1 : hl=1 :
     h5=pg(14)
22871 gosub 22790 : h3=int(xh%/10)/10 :
     h4=int(yh%/10)/10
22872 while il : gosub 22390 : gosub 22350 : line input
     "Text? ";n$ : il=len(n$) : if il=0 then 22878
22873   if mid$(n$,1,2)="&A" then n$=n$+"  " :
     pg(7)=val(mid$(n$,3,2)) : goto 22878
22874   if mid$(n$,1,2)="&R" then n$=n$+"  " :
     hl=val(mid$(n$,3,1)) : goto 22878
```

```
22875   if mid$(n$,1,2)="&S" then n$=n$+"  " : goto
     22878
22876   pg(14)=pg(14)+1 : i=pg(14)+15 : pg(i)=h1 :
     pgx(i)=h3 : pgy(i)=h4 : if h1=1 then h4=h4-pg(7)
     else h3=h3-pg(7)
22877   beschr$(pg(14)+5)=n$+space$(80-il) :
     tl(pg(14)+5)=il : if pg(14)>14 then il=0 : goto
     22878
22878 wend  : gosub 22390
22879 for i=81 to 84 : ip(i)=1 : next i : print gt$; :
     gosub 23090 : return
22880 gosub 23090 : pg(14)=0 : return
22890 gosub 22390 : print gt$; : pointer%(22)=0 :
     block%=2 : gosub 22030 : gosub 22350 : return
22895 pointer%(22)=1 : block%=2 : gosub 22030 : gosub
22390 : return
rem *************** More softkeys *********************
22900 ic=1 : gosub 28100 : if a<0 or a>9 then return
22902 on a gosub
22910,22920,22930,22940,22950,22960,22970,22980
22905 return
22910 gosub 22390 : gosub 22350 : input "Curve No.(1-
     5), smoothtype (1-3)";j,smooth : if j=0 or
     smooth=0 then gosub 22390 : return
22913 if j>10 or j<1 or smooth>3 or smooth<1 then goto
     22910
22915 gosub 23300 : gosub 22390 : gosub 27500 : gosub
23300 : return : rem smoothen
22920 gosub 22390 : gosub 22350 : input "Select the
     curve to interpolate (1-10)",j : if j=0 then gosub
22390 : return
22923 if j>10 or j<1 then goto 22920
22925 gosub 23300 : gosub 22390 : gosub 26500 : gosub
23300 : return : rem interpolieren
```

```
22930 gosub 22390 : gosub 22350 : input "Please select
     curve (1-10)";j : if j=0 then gosub 22390 : return
22933 if j>10 or j<1 goto 22930
22935 gosub 23300 : gosub 22390 : psym(j)=insym : gosub
23300 : return : rem interpolieren
22940 gosub 22390 : gosub 22350 : input "Curve (1-10),
     pen (1-8)";j,ik : if j<1 or j>10 then gosub 22390
     : return
22945 gosub 22390 : if stift%(j)<>0 and ik=0 or
     stift%(j)=0 and ik<>0 then gosub 23300
22947 stift%(j)=ik : return
22950 for block%=1 to 9 : gosub 22030 : next block% :
     pointer%(23)=1 : gosub 22100 : return
22960 call gclear : for i=91 to 99 : ip(i)=0 : next i :
     gosub 22000 : return
22970 for i=91 to 99 : ip(i)=0 : next i : if
     pointer%(21)=0 then pointer%(21)=1 else
     pointer%(21)=0
22975 call gclear : gosub 22000 : return
22980 tsset=1 : gosub 22100 : return
22990 return
rem  ****************** Display on screen *************
rem
rem ******** display clear and set scale **************
23000 tshare=4 : on error goto 21900 : gosub 23600
23005 pxmin=60 : pxmax= 460 : pymin=90 : pymax=340 : if
     pointer%(21)<>1 then 23025
23010 pxmin=pg(3)*19 : pxmax=(pg(3)+pg(1))*19 :
     pymin=pg(4)*18+32 : pymax=(pg(2)+pg(4))*18+32
23015 if pg(13) mod 2=0 then pxmin=pg(4)*15+100 :
     pxmax=(pg(4)+pg(1))*15+100 : pymin=390-
     (pg(2)+pg(3))*13 : pymax=390-pg(3)*13
23020 goto 23030
```

```
23025 h4=pg(2)/pg(1)*1.6 : proport=h4 : if h4>1 then
     pxmax=int(pxmin+400/h4) else
     pymax=int(pymin+250*h4)
23030 if pg(12)=0 then goto 23045
23035 if pointer%(21)<>1 then call gclear : for i=91 to
     99 : ip(i)=0 : next i
23040 if pointer%(24)=1 then print chr$(27)+"*m1G"; :
     if pg(13) mod 2=0 and pointer%(21)=1 then print
     chr$(27)+"*m 100 30 391 389 E"; else print
     chr$(27)+"*m 0 30 511 389 E";
23045 call graph : call gcurs : return
rem ******** display axes *****************************
23050 si=pgx(3) : siy=pgy(3) : gosub 29100 : rem call
     size
23052 print gt$;
23055 scxf#=fnscx(pxu#(1),pxu#(2)) : scyf#=fnscy
     (pyl#(1),pyl#(2)) : scadx#=pxu#(1) :
     scady#=pyl#(1)
23060 f5=(pxu#(2)-pxu#(1))/100 : f6=(pyl#(2)-
     pyl#(1))/100 : if pg(12)=0 then goto 23250
23061 aypos=pyl#(1) : axpos=pxu#(1)
23062 min#=pxu#(1) : max#=pxu#(2) : del#=pxu#(3) :
     delo#=pxu#(4) : d%=1 : call dir(d%)
23064 if pxu#(5)>0 then zxpo=pgx(12) : zypo=pgy(12) :
     ptk=f6 : aop=pxu#(5) : gosub 25900 : rem x-achse
23065 if pxo#(5)<0 then aypos=pyl#(2) : zxpo=pgx(14) :
     zypo=pgy(14) : ptk=-f6 : aop=pxo#(5) : gosub 25900
23066 min#=pyl#(1) : max#=pyl#(2) : del#=pyl#(3) :
     delo#=pyl#(4)
23068 if pyl#(5)>0 then zxpo=pgx(13) : zypo=pgy(13) :
     ptk=f5 : aop=pyl#(5) : gosub 25950 : rem y-achse
23069 if pyr#(5)<0 then axpos=pxu#(2) : zxpo=pgx(15) :
     zypo=pgy(15) : ptk=-f5 : aop=pyr#(5) : gosub 25950
```

```
23070 scxf#=fnscx(pxo#(1),pxo#(2)) : f5=(pxo#(2)-
     pxo#(1))/100 : scadx#=pxo#(1) : axpos=pxo#(2)
23072 scyf#=fnscy(pyr#(1),pyr#(2)) : f6=(pyr#(2)-
     pyr#(1))/100 : scady#=pyr#(1) : aypos=pyr#(2)
23074 min#=pxo#(1) : max#=pxo#(2) : del#=pxo#(3) :
     delo#=pxo#(4) : aop=pxu#(5)
23076 if pxo#(5)>0 then zxpo=pgx(14) ‖ zypo=pgy(14) ‖
     ptk=-f6 ‖ aop=pxo#(5⌐ : gosub 25900 : rem x-achse
23077 if pxu#(5)<0 then aypos=pyr#(1) : zxpo=pgx(12) :
     zypo=pgy(12) : ptk=f6 : aop=pxu#(5) : gosub 25900
23078 min#=pyr#(1) : max#=pyr#(2) : del#=pyr#(3) :
     delo#=pyr#(4) : aop=pyr#(5)
23080 if pyr#(5)>0 then zxpo=pgx(15) : zypo=pgy(15) :
     ptk=-f5 : aop=pyr#(5) : gosub 25950 : rem y-achse
23081 if pyl#(5)<0 then axpos=pxo#(1) : zxpo=pgx(13) :
     zypo=pgy(13) : ptk=f5 : aop=pyl#(5) : gosub 25950
23082 scxf#=fnscx(pxu#(1),pxu#(2)) :
     scyf#=fnscy(pyl#(1),pyl#(2)) : scadx#=pxu#(1) :
     scady#=pyl#(1)
23084 return
rem ****************** display annotate **************
23090 if pg(14)=0 or pg(15)=0 then goto 23099
23091 scxf#=fnscx(0,100) : scyf#=fnscy(0,100) :
     scadx#=0 : scady#=0 : print togglem$; : print gt$;
23092 si=pgx(5) : siy=pgy(5) : gosub 29100 : print gt$;
23093 for i=1 to pg(14) : d%=pg(i+15) : if d%>4 or d%<1
     then d%=1
23094 call dir(d%) : pdx%=fnxw(pgx(i+15)) :
     pdy%=fnyw(pgy(i+15)) : if pdx%>511 or pdx%<0 or
     pdy%>389 or pdy%<0 then goto 23097
23096 call plotu(pdx%,pdy%) : print mid$(beschr$
     (i+5),1,tl(i+5))
23097 next i
23099 return
```

```
rem ************ display frame and write title ********
23100 scxf#=fnscx(0,100) : scyf#=fnscy(0,100) :
     scadx#=0 : scady#=0 : print togglem$;
23101 si=pgx(2) : siy=pgy(2) : gosub 29100 : print gt$;
     : if pg(12)=0 then 23120
23102 f3=si*100/(pxmax-pxmin) : f4=si*100/(pymax-pymin)
     : sl=-(int(len(xbes$))/2) : xh%=fnxw(sl*f3+pgx(7))
     : yh%=fnyw(pgy(7)) : call plotu(xh%,yh%)
23104 if pxu#(5) mod 8>3 then print xbes$
23109 if pxo#(5) mod 8>3 then sl=-(int(len(xobes$))/2)
     : xh%=fnxw(sl*f3+pgx(9)) : yh%=fnyw(pgy(9)) : call
     plotu(xh%,yh%) : print xobes$
23110 if pyl#(5) mod 8>3 then sl=-(int(len(ybes$))/2) :
     d%=2 : call dir(d%) : xh%=fnxw(pgx(8)) :
     yh%=fnyw(sl*f4+pgy(8)) : call plotu(xh%,yh%) :
     print ybes$ : d%=1 : call dir(d%)
23112 if pyr#(5) mod 8>3 then sl=-(int(len(yrbes$))/2)
     : d%=2 : call dir(d%) : xh%=fnxw(pgx(10)) :
     yh%=fnyw(sl*f4+pgy(10)) : call plotu(xh%,yh%) :
     print yrbes$ : d%=1 : call dir(d%)
23115 l%=1 : call linetype(l%) : xh%=fnxw(0) :
     yh%=fnyw(0) : call plotu(xh%,yh%)
23116 xh%=fnxw(0) : yh%=fnyw(100) : call plotd(xh%,yh%)
23117 xh%=fnxw(100) : yh%=fnyw(100) : call
     plotd(xh%,yh%)
23118 xh%=fnxw(100) : yh%=fnyw(0) : call plotd(xh%,yh%)
23119 xh%=fnxw(0) : yh%=fnyw(0) : call plotd(xh%,yh%) :
     call penup
23120 si=pgx(1) : siy=pgy(1) : gosub 29100 : if
     pg(11)=0 then 23125
23122 f3=si*100/(pxmax-pxmin) : d%=0 : call dir(d%) :
     sl=-int((len(ptitel$))/2) : xh%=fnxw(pgx(6)+f3*sl)
     : yh%=fnyw(pgy(6)) : call plotu(xh%,yh%) : print
     ptitel$
```

```
23125 return
rem ************display data**********************
23250 scxf#=fnscx(pxu#(1),pxu#(2)) : scyf#=fnscy(py
     l#(1),pyl#(2)) : scadx#=pxu#(1) : scady#=pyl#(1)
23270 for j=1 to 10 : if stift%(j)>0 then gosub 5000 :
     gosub 23300
23275 next j
23290 return
23300 isym=psym(j) : si=pgx(4) : siy=pgy(4) : gosub
29100 : rem call size
23305 if dmax(j)=5 or psym(j)<1 then goto 23395
23307 if dmax(j)>320 then dmax(j)=320
23310 if isym>40 then isym=1
23320 if isym>0 and isym<11 then if pointer%(25)<>1
     then i=1 : gosub 26400
23330 if isym<11 then goto 23350 else gosub 26100 : rem
     define Linie
23332 if dx(j,4)>pxu#(2) or dx(j,4)<pxu#(1) or
     dy(j,4)>pyl#(2) or dy(j,4)<pyl#(1) then xh%=pxmin
     : yh%=pymin : goto 23350
23340 xh%=fnxw(dx(j,4)) : yh%=fnyw(dy(j,4)) : call
     penup : call plotu(xh%,yh%)
23350 for i=4 to dmax(j) : jf=0
23352     if dx(j,i)=99999 or dx(j,i)<pxu#(1) or
     dx(j,i)>pxu#(2) or dy(j,i)>pyl#(2) or
     dy(j,i)<pyl#(1) then call penup : goto 23380
23354     if dx(j,i-1)=99999 or dx(j,i-1)<pxu#(1) or
     dx(j,i-1)>pxu#(2) or dy(j,i-1)>pyl#(2) or dy(j,i-
     1)<pyl#(1) then jf=1
23360     xh%=fnxw(dx(j,i)) : yh%=fnyw(dy(j,i)) : if
     xh%>511 then xh%=511 : call penup : call
     plotu(xh%,yh%) : goto 23380
23368     if xh%<0 then xh%=0 : call penup : call
     plotu(xh%,yh%) : goto 23380
```

```
23369     if yh%>389 then yh%=389 : call penup : call
     plotu(xh%,yh%) : goto 23380
23370     if yh%<0 then yh%=0 : call penup : call
     plotu(xh%,yh%) : goto 23380
23371     if dx(j,i-1)=99999 then call penup : call
     plotu(xh%,yh%)
23372     if jf=1 then call plotu(xh%,yh%) : goto 23375
23373     if isym>10 and isym<21 then call
     plotd(xh%,yh%) : goto 23380
23375     if isym<11 then call plotu(xh%,yh%) : print
     chr$(27)+sym$(isym); : goto 23380
23377     if isym>20 and isym<31 then x1=dx(j,i) :
     y1=dy(j,i) : gosub 23800 : goto 23380
23378     if isym>30 and isym<41 then call
     plotu(xh%,yh%) : h5$=mid$(zeichen$,isym-30,1) :
     print h5$
23380 next i
23385 if psym(j)>10 then call penup
23395 return
23600 rem *********check axes *************************
23605 if pxu#(3)=0 or pyl#(3)=0 or pxo#(3)=0 or
     pyr#(3)=0 then 23690
23610 if pxu#(2)-pxu#(1)<=0 or pyl#(2)-pyl#(1)<=0  then
     goto 23690
23615 if pxo#(5)>0 then if pxo#(2)-pxo#(1)=0 or
     (pxo#(2)-pxo#(1))/pxo#(3)>99 then goto 23690
23620 if pyr#(5)>0 then if pyr#(2)-pyr#(1)=0 or
     (pyr#(2)-pyr#(1))/pyr#(3)>99 then goto 23690
23625 if (pxu#(2)-pxu#(1))/pxu#(3)>99 then goto 23690
23630 if (pyl#(2)-pyl#(1))/pyl#(3)>99 then goto 23690
23680 return
23690 print "****illegal axis******" : for i=1 to 4000
     : k=i : next i : goto 500
rem **************Line Drawing*************************
```

```
23700 if idmax<2 then goto 23760
23710 l%=1 : call linetype(l%) : xh%=idx(1) :
      yh%=idy(1) : gosub 22795 : call plotu(xh%,yh%)
23720 for i=1 to idmax : xh%=idx(i) : yh%=idy(i) :
      gosub 22795 : if idp(i)>100 then gosub 23770 :
      goto 23750
23725 if idp(i)>10 then l%=idp(i)-10 : call
      linetype(l%)
23730 if idp(i)=0 then call penup : call plotu(xh%,yh%)
      else call plotd(xh%,yh%)
23750 next i
23760 return
23770 isym=idp(i)-100 : call plotu(xh%,yh%) : if
      isym<10 then print sym$(isym)
23775 if isym>20 and isym<31 then x1=idx(i)/100 :
      y1=idy(i)/100 : gosub 23800
23780 return
rem ****************Columns Drawing *****************
23800 l%=1 : call linetype(l%) : call penup
23805 x2=(pxu#(2)-pxu#(1))/200 : y2=(pyl#(2)-
      pyl#(1))/200 : y0=pyl#(1)
23810 x=x1-x2*pga(8) : y=y0 : gosub 23995 : xul%=xh% :
      yul%=yh% : y=y1 : gosub 23990 : x=x1+x2*pga(8) :
      gosub 23990 : ixor=xh% : yor%=yh%
23815 y=y0 : gosub 23990 : x=x1-x2*pga(8) : gosub 23990
      : call penup
23820 if isym=21 then goto 23890
23825 if isym=22 then gosub 23900 : goto 23890
23830 if isym=23 then gosub 23920 : goto 23890
23835 if isym=24 then gosub 23900 : gosub 23920 : goto
23890
23840 if isym=25 then print chr$(27);"*m 128 64 32 16 8
      4 2 1 D"; : print chr$(27);"*m2G"; : print
```

```
     chr$(27);"*m";str$(xul%);str$(yul%);str$(ixor);str
     $(yor%);"E";
23845 if isym=26 then print chr$(27);"*m 1 2 4 8 16 32
     64 128 D"; : print chr$(27);"*m2G"; : print
chr$(27);"*m";str$(xul%);str$(yul%);str$(ixor);str$(yor
     %);"E";
23850 if isym=27 then print chr$(27);"*m 129 66 36 24
     24 36 66 129 D"; : print chr$(27);"*m2G"; : print
     chr$(27);"*m";str$(xul%);str$(yul%);str$(ixor);str
     $(yor%);"E";
23855 if isym=28 then print chr$(27);"*m 255 255 255
     255 255 255 255 255 D"; : print chr$(27);"*m2G"; :
     print chr$(27);"*m";str$(xul%);
     str$(yul%);str$(ixor);str$(yor%);"E";
23860 if isym=29 then l%=7 : call linetype(l%) : gosub
23900 : goto 23890
23865 if isym=30 then gosub 23940 : goto 23890
23890 return
23900 for h1=x1-x2*pga(8) to x1+x2*pga(8) step x2 :
     x=h1 : y=y0 : gosub 23995 : y=y1 : gosub 23990 :
     call penup : next h1
23905 return
23920 for h1=y0 to y1 step 2*y2 : x=x1-pga(8)*x2 : y=h1
     : gosub 23995 : x=x1+pga(8)*x2 : gosub 23990 :
     call penup  : next h1
23925 return
23940 for h1=x1-x2*pga(8) to x1+x2*pga(8) step x2/4 :
     x=h1 : y=y0 : gosub 23995 : y=y1 : gosub 23990 :
     call penup : next h1
23945 return
23990 xh%=fnxw(x) : yh%=fnyw(y) : call plotd(xh%,yh%) :
     return
23995 xh%=fnxw(x) : yh%=fnyw(y) : call plotu(xh%,yh%) :
     return
```

```
rem  ****************** Print table on printer ********
rem
rem **************print header ***********************
24000 tshare=1 : print#11,ptitel$ : print#11,"  " :
     gosub 5000
24010 print#11,"X-axis  : ";xbes$;"          Y-axis  :
     ";ybes$ : gosub 5000 : gosub 5000
24020 print#11,"X-axis top   : ";xobes$;"          Y-
     axis right   : ";yrbes$ : print#11,"     " : gosub
     5000
24040 if pg(14)=0 then goto 24080
24050 for i=1 to pg(14) : gosub 5000 : print#11,mid$
     (beschr$(i+5),1,tl(i+5)) : next i : print#11,"  "
rem ****************print data*******************
24080 rem formp1$="   ####    +####.####   +####.####
     " : textp1$=" Point No.   X-value    Y-value     "
24090 for j=1 to 10 : if dmax(j)<5 then goto 24500
24100   print#11,"Curve  ";str$(j);"  :" :
     print#11,string$(79,95) : print#11,"     " :
     print#11,format$(86);format$(86) : print#11,"    "
24190   iffs=int((dmax(j)-4)/2)+1
24200   for i=1 to int((dmax(j)-4)/2)+1 : i1=i+iffs :
     form$=format$(85)+format$(85)
24205     for iji=1 to 20 : gosub 5000 : if a$="s" then
24500
24210       if a$="a" then return
24215     next iji
24220     h1=abs(dx(j,i+3)) : if h1>9999 or h1<0.01
     then mid$(form$,11,11)="+#.####^^^^"
24230     h1=abs(dy(j,i+3)) : if h1>9999 or h1<0.01
     then mid$(form$,24,11)="+#.####^^^^"
24240     h1=abs(dx(j,i1+3)) : if h1>9999 or h1<0.01
     then mid$(form$,51,11)="+#.####^^^^"
```

```
24250     h1=abs(dy(j,i1+3)) : if h1>9999 or h1<0.01
     then mid$(form$,64,11)="+#.####^^^^"
24260     if i1+3>dmax(j) then print#11, using
     mid$(form$,1,39);i,dx(j,i+3),dy(j,i+3)
24270     if i1+3<=dmax(j) then print#11, using
     form$;i,dx(j,i+3),dy(j,i+3),i1,dx(j,i1+3),dy(j,i1+
     3)
24300   next i : print#11,string$(79,95) : print#11,"
     "
24500 next j : return
rem  ****************** Plot data on plotter ***********
rem
rem ***************plotter setup***********************
25000 gosub 23600
25005 on error goto 6000 : aufspulen%=0 : close#1 :
     open "o",#1,"PLT" : width #1,128
25010 tshare=2
25020 pxmin=pg(3)*400 : pxmax=(pg(3)+pg(1))*400 :
     pymin=pg(4)*400 : pymax=(pg(4)+pg(2))*400 :
     pzmax=(pg(3)+pg(2))*400
25030 scxf#=fnscx(0,100) : scyf#=fnscy(0,100) :
     scadx#=0 : scady#=0
25035 pl$="pu;pa 0,0;" : gosub 25800 : pl$="in;ip
     0,0,10900,7650;" : gosub 25800
25040 pl$="vs"+str$(pg(5))+";" : gosub 25800 : if
     pg(12)=0 then goto 25150 else pl$="sp"+str
     $(pg(12))+";" : gosub 25800
25050 ipen=0 : h1=0 : h2=0 : gosub 26000 : ipen=1 :
     h2=100 : gosub 26000 : h1=100 : gosub 26000 : h2=0
     : gosub 26000 : h1=0 : gosub 26000
25060 h1=0.1 : gosub 26000 : h2=100.1 : gosub 26000 :
     h1=100.1 : gosub 26000 : h2=0.1 : gosub 26000 :
     h1=0.1 : gosub 26000 : ipen=0 : gosub 26000
```

```
25100 sl=-(int(len(ptitel$)/2)) : pl$="sp"+str$(pg
      (11))+";si"+str$(pgx(1))+","+str$(pgy(1))+";" :
      gosub 25800
25105 aktcpy=0 : aktsix=pgx(1) : aktsiy=pgy(1)
25110 h1=pgx(6) : h2=pgy(6) : ipen=0 : gosub 26000 : if
      pg(13) mod 2=0 then pl$=pl$+"di 0,1;" else pl$=""
25115 pl$=pl$+"cp"+str$(sl)+",0;" : gosub 25800 :
      pl$=ptitel$ : gosub 25820 : if pg(11)<>pg(12) then
      pl$="sp"+str$(pg(12))+";" : gosub 25800
25116 aktcpy=0 : aktsix=pgx(2) : aktsiy=pgy(2)
25117 sl=-(int(len(xbes$)/2)) : pl$="si"+str$
      (pgx(2))+","+str$(pgy(2))+";" : gosub 25800 : if
      pxu#(5) mod 8<4 then goto 25127
25120 h1=pgx(7) : h2=pgy(7) : ipen=0 : gosub 26000 : if
      pg(13) mod 2=0 then pl$="di 0,1;" else pl$=""
25126 pl$=pl$+"cp"+str$(sl)+",0;" : gosub 25800 :
      pl$=xbes$ : gosub 25820
25127 sl=-(int(len(ybes$)/2)) : if pyl#(5) mod 8<4 then
      goto 25132
25128 h1=pgx(8) : h2=pgy(8) : ipen=0 : gosub 26000 : if
      pg(13) mod 2=0 then pl$="di -1,0;" else pl$="di
      0,1;"
25131 pl$=pl$+"cp"+str$(sl)+",0;" : gosub 25800 :
      pl$=ybes$ : gosub 25820
25132 pl$="di 1,0;" : gosub 25800 : sl=-(int(len
      (yrbes$)/2)) : if pyr#(5) mod 8<4 then goto 25137
      25133 h1=pgx(10) : h2=pgy(10) : ipen=0 : gosub
      26000 : if pg(13) mod 2=0 then pl$="di -1,0;" else
      pl$="di 0,1;"
25135 pl$=pl$+"cp"+str$(sl)+",0;" : gosub 25800 :
      pl$=yrbes$ : gosub 25820
25137 pl$="di 1,0;" : gosub 25800 : sl=-(int(len(xobes
      $)/2)) : if pxo#(5) mod 8<4 then goto 25150
```

```
25140 h1=pgx(9) : h2=pgy(9) : ipen=0 : gosub 26000      :
     if pg(13) mod 2=0 then pl$=pl$+"di 0,1;" else
     pl$=""
25144 pl$=pl$+"cp"+str$(sl)+",0;" : gosub 25800 :
     pl$=xobes$ : gosub 25820
rem
******************annotate*************************
25150 if pg(14)=0 or pg(15)=0 then goto 25192
25156 aktcpy=0 : aktsix=pgx(5) : aktsiy=pgy(5)
25157 pl$="si"+str$(pgx(5))+","+str$(pgy(5))+";
     sp"+str$(pg(15))+";" : gosub 25800
25160 for i=1 to pg(14) : pl$="pu;" : gosub 25800 :
     h1=pgx(i+15) : h2=pgy(i+15) : ipen=0 : gosub 26000
25165 if pg(13) mod 2=1 then if pg(i+15)=1 then pl$="cp
     0,0;di 1,0;" else pl$="cp 0,0;di 0,1;"
25170 if pg(13) mod 2=0 then if pg(i+15)=1 then pl$="cp
     0,0;di 0,1;" else pl$="cp 0,0;di -1,0;"
25172 gosub 25800 : pl$=mid$(beschr$(i+5),1,tl(i+5)) :
     gosub 25820
25190 next i
rem
****************axes*********************************
25192 if pg(15)<>0 then gosub 25850
25195 if pg(12)=0 then goto 25400 else
     pl$="sp"+str$(pg(12))+";" : gosub 25800
25200 scxf#=fnscx(pxu#(1),pxu#(2)) : scyf#=fnscy(pyl#
     (1),pyl#(2)) : scadx#=pxu#(1) : scady#=pyl#(1)
25205 f5=(pxu#(2)-pxu#(1))/100 : f6=(pyl#(2)-
     pyl#(1))/100
25210 pl$="si"+str$(pgx(3))+","+str$(pgy(3))+";" :
     gosub 25800
25215 min#=pxu#(1) : max#=pxu#(2) : del#=pxu#(3) :
     delo#=pxu#(4)
```

```
25220 aypos=pyl#(1) : zxpo=pgx(12) : zypo=pgy(12) :
     ptk=f6 : aop=pxu#(5) : gosub 25900 : rem x-achse
25250 min#=pyl#(1) : max#=pyl#(2) : del#=pyl#(3) :
     delo#=pyl#(4)
25260 axpos=pxu#(1) : zxpo=pgx(13) : zypo=pgy(13) :
     ptk=f5 : aop=pyl#(5) : gosub 25950 : rem y-achse
25280 if pxo#(5)>-1 then scxf#=fnscx(pxo#(1),pxo#(2)) :
     f5=(pxo#(2)-pxo#(1))/100 : scadx#=pxo#(1) :
     axpos=pxo#(2) else axpos=pxu#(2)
25290 if pyr#(5)>-1 then scyf#=fnscy(pyr#(1),pyr#(2)) :
     f6=(pyr#(2)-pyr#(1))/100 : scady#=pyr#(1) :
     aypos=pyr#(2) else aypos=pyl#(2)
25300 if pxo#(5)<0 then min#=pxu#(1) : max#=pxu#(2) :
     del#=pxu#(3) : delo#=pxu#(4) else min#=pxo#(1) :
     max#=pxo#(2) : del#=pxo#(3) : delo#=pxo#(4) :
     aop=pxu#(5)
25320 zxpo=pgx(14) : zypo=pgy(14) : ptk=-f6 :
     aop=pxo#(5) : gosub 25900 : rem x-achse
25340 if pyr#(5)<0 then min#=pyl#(1) : max#=pyl#(2) :
     del#=pyl#(3) : delo#=pyl#(4) else min#=pyr#(1) :
     max#=pyr#(2) : del#=pyr#(3) : delo#=pyr#(4) :
     aop=pyr#(5)
25350 zxpo=pgx(15) : zypo=pgy(15) : ptk=-f5 :
     aop=pyr#(5) : gosub 25950 : rem y-achse
25400 scxf#=fnscx(pxu#(1),pxu#(2)) : scyf#=fnscy
     (pyl#(1),pyl#(2)) : scadx#=pxu#(1) :
     scady#=pyl#(1)
25600 rem **************plot of data ******************
25601 if pg(13)=1 or pg(13) mod 2=0 then pl$="iw"+fnw$
     (pxu#(1),pyl#(1))+","+fnw$(pxu#(2),pyl#(2))+";"
     else  pl$="iw"+fnv$(pxu#(1),pyl#(1))+",
     "+fnv$(pxu#(2),pyl#(2))+";"
25605 pl$="si"+str$(pgx(4))+","+str$(pgy(4))+";
     sp"+str$(stift%(1))+";cp 0,0;" : gosub 25800
```

```
25610 for j=1 to 10  : isym=psym(j) : if stift%(j)=0 or
     dmax(j)<5 or isym<1 then 25770
25620   if j>1 and stift%(j)<>stift%(j-1) then
     pl$="sp"+str$(stift%(j))+";" : gosub 25800
25630   if isym<11 and pointer%(25)<>3 then i=3 : gosub
     26400
25650   gosub 26100 : if isym>10 then ipen=0 :
     h1=dx(j,4) : h2=dy(j,4) : gosub 26000
25670   for i=4 to dmax(j) : h1=dx(j,i) : h2=dy(j,i) :
     if i mod 5=0 then gosub 5000
25672     if a$="a" then gosub 25780 : goto 500
25674     if a$="s" then ipen=0 : gosub 26000 : goto
25750
25680     if dx(j,i)>pxu#(2) or dx(j,i)<pxu#(1) or
     dy(j,i)>pyl#(2) or dy(j,i)<pyl#(1) then  ipen=0 :
     h1=dx(j,i+1) : h2=dy(j,i+1) : gosub 26000 : goto
25750
25690     if i>4 then if dx(j,i-1)>pxu#(2)+abs(pxu#(1))
     or dx(j,i-1)<pxu#(1)-abs(pxu#(2)) or dy(j,i-
     1)>pyl#(2)+abs(pyl#(1)) or dy(j,i-1)<pyl#(1)-
     abs(pyl#(2)) then : goto 25750
25730     if isym>10 and isym<21 then ipen=1 : gosub
     26000 : goto 25750
25740     if isym<11 then pl$="pu;" : gosub 25800 :
     ipen=0 : gosub 26000 : pl$=sym$(isym) : gosub
     25800 : goto 25750
25745     if isym>20 and isym<31 then x1=dx(j,i) :
     y1=dy(j,i) : gosub 26800 : goto 25750
25747     if isym>30 and isym<41 then ipen=0 : gosub
     26000 : h5$=mid$(zeichen$,isym-30,1) : pl$=h5$ :
     gosub 25820
25750   next i
25760   if isym>10 then h1=dx(j,dmax(j)) :
     h2=dy(j,dmax(j)) : ipen=0 : gosub 26000
```

```
25770 next j
25780 pl$="pu;pa 0,0;sp 0;" : gosub 25800
25785 out$="Done" : close#1
25790 return
25800 print#1,chr$(34);pl$;chr$(34) : return
25820 h5$=pl$ : for iii=1 to len(h5$) :
     h6$=mid$(h5$,iii,1) : ichar=asc(h6$)
25821    if h6$="@" then cyoff=0.3 : gosub 25830 : goto
     25829
25822    if h6$="#" then cyoff=-0.3 : gosub 25830 :
     goto 25829
25823    if h6$="&" then iii=iii+1 : ichar=asc(mid$
     (h5$,iii,1))-48 : gosub 25835 : goto 25829
25824    for iiii=1 to 20 : if ichar=texept(iiii) then
     ichar=iiii : gosub 25835 : goto 25829
25825    next iiii
25826 pl$="lb"+mid$(h5$,iii,1)+chr$(3)+";" : gosub
25800
25829 next iii : return
25830 iii=iii+1 : pl$="cp 0,"+str$(cyoff)+";si"+
     str$(aktsix*.75)+","+str$(aktsiy*.75)+";lb"+mid$(h
     5$,iii,1)+chr$(3)+";" : gosub 25800
25832 pl$="si"+str$(aktsix)+","+str$(aktsiy)+";cp
     0,"+str$(-cyoff)+";" : gosub 25800 : return
25835 if ichar>1 and ichar<11 and pointer%(25)<>4 then
     i=4 : gosub 26400
25836 if ichar>10 and ichar<21 and pointer%(25)<>4 then
     i=4 : gosub 26400 : ichar=ichar-10
25837 if ichar>1 and ichar<11 then pl$=sym$(ichar) :
     gosub 25800
25839 return
rem **************Line Drawing*********************
25850 if idmax<2 then goto 25890
25855 pl$="lt;" : gosub 25800
```

```
25860 for i=1 to idmax : h1=idx(i)/100 : h2=idy(i)/100
     : if i mod 5=0 then gosub 5000
25865 if idp(i)>100 then gosub 25895 : goto 25885
25870 if idp(i)>10 then l%=idp(i)-10 : call
     linetype(l%)
25875 if idp(i)=0 then ipen=0 : gosub 26000
25880 if idp(i)=1 then ipen=1 : gosub 26000
25885 next i
25890 return
25895 isym=idp(i)-100 : if isym<11 then ipen=0 : gosub
26000 : pl$=sym$(isym) : gosub 25800
25897 if isym>20 and isym<31 then x1=h1 : y1=h2 : gosub
26800
25899 return
25900 rem *****X-Axis **********************************
25902 if ttaus=1 then zypo=zypo-.35 : f3=si*f5*100/
     (pxmax-pxmin) : f4=siy*f6*100/(pymax-pymin)
25905 ih=0 : ap%=abs(aop) : ax%=ap% mod 4 : if ax%=1 or
     ax%=3 then at%=1 else at%=0
25906 if ax%=2 or ax%=3 then az%=1 else az%=0
25907 if ap%>7 then lg%=1 else lg%=0
25908 if ax%=0 or del#=0 or (max#-min#)/del#>99 then
     goto 25940
25910 for k#=min#+delo# to max# step del# : if lg%=0
     then k=k# : k1=k : goto 25916
25912 ih=int((k#-int(k#))*100) : if ih<=0 then
     h1#=10^int(k#) else h1#=10^(int(k#) +log(ih/10)/
     log(10))
25913 if h1#>0 then h#=log(h1#)/log(10) else h1#=1
25914 k=h# : k1=h1# : if k1=0 then k1=1
25915 xx=k#*pga(9)-cint(k#*pga(9)) : if abs(xx)<0.0001
     then ih=1 else ih=0
25916 if ttaus=1 then 25918 else gosub 5000 : if a$="a"
     then gosub 25780 : goto 500
```

```
25917 if a$="s" then 25935
25918 if at%=0 then goto 25925
25920 h1=k : h2=aypos : ipen=0 : gosub 26000 : ipen=1 :
      h2=h2+ptk*pg(6) : gosub 26000
25925 if az%=0 then goto 25935
25927 if lg%=1 and ih=0 then 25935
25929 if lg%=0 and abs(k#)<abs(del#)/1000000 then k1=0
25930 ipen=0 : if ttaus=1 then h1=k+zxpo*f3  :
      h2=aypos+zypo*f4 : gosub 26000 : print str$(k1) :
      goto 25935
25931 h1=k : h2=aypos : gosub 26000 : if pg(13) mod 2=0
      then pl$="di 0,1;" else pl$="di 1,0;"
25932 pl$=pl$+"cp"+str$(zxpo)+","+str$(zypo)+";" :
      gosub 25800 : pl$=str$(k1) : gosub 25820
25935 next k#
25940 return
25950 rem *********Y-Axis *****************************
25952 if ttaus=1 then zypo=zypo-.35 : f3=si*f5*100/
      (pxmax-pxmin) : f4=siy*f6*100/(pymax-pymin)
25955 ih=0 : ap%=abs(aop) : ax%=ap% mod 4 : if ax%=1 or
      ax%=3 then at%=1 else at%=0
25956 if ax%=2 or ax%=3 then az%=1 else az%=0
25957 if ap%>7 then lg%=1 else lg%=0
25958 if ax%=0 or del#=0 or (max#-min#)/del#>99 then
      goto 25990
25960 for k#=min#+delo# to max# step del# : if lg%=0
      then k=k# : k1=k : goto 25966
25962 ih=int((k#-int(k#))*100) : if ih<=0 then
      h1#=10^int(k#) else h1#=10^(int(k#)+log
      (ih/10)/log(10))
25963 if h1#>0 then h#=log(h1#)/log(10) else h1#=1
25964 k=h# : k1=h1# : if k1=0 then k1=1
25965 xx=k#*pga(9)-cint(k#*pga(9)) : if abs(xx)<0.0001
      then ih=1 else ih=0
```

```
25966 if ttaus=1 then 25968 else gosub 5000 : if a$="a"
     then gosub 25780 : goto 500
25967 if a$="s" then 25985
25968 if at%=0 then goto 25975
25970 h1=axpos : h2=k : ipen=0 : gosub 26000 : ipen=1 :
     h1=h1+ptk*pg(6) : gosub 26000
25975 if az%=0 then goto 25985
25977 if lg%=1 and ih=0 then 25985
25979 if lg%=0 and abs(k#)<abs(del#)/1000000 then k1=0
25980 ipen=0 : if ttaus=1 then h1=axpos+zxpo*f3 :
     h2=k+zypo*f4 : gosub 26000 : print str$(k1) : goto
     25985
25981 h1=axpos : h2=k : gosub 26000 : if pg(13) mod 2=0
     then pl$="di 0,1;" else pl$="di 1,0;"
25982 pl$=pl$+"cp"+str$(zxpo)+","+str$(zypo) : gosub
25800 : pl$=str$(k1) : gosub 25820
25985 next k#
25990 return
rem ****************plot absolute + transformation*****
26000 if ttaus=1 then 26050
26002 if ttaus>2 then return
26005 if ipen=1 then 26025
26010 if pg(13)=1 or pg(13)=3 then pl$=fnpmu$(h1,h2)
     else pl$=fnpmu2$(h1,h2)
26020 gosub 25800 : return
26025 if pg(13)=1 or pg(13)=3 then pl$=fnpm$(h1,h2)
     else pl$=fnpm2$(h1,h2)
26030 gosub 25800 : return
26050 x%=fnxw(h1) : y%=fnyw(h2) : if ipen=1 then call
     plotd(x%,y%) else call plotu(x%,y%)
26055 return
rem *******************linie type********************
26100 if ttaus=1 then 26150
```

```
26110 l%=isym-11 : if isym=11 or isym>20 then pl$="lt;"
     : goto 26140
26120 if isym>13 then pl$="lt"+str$(l%)+",3;" : goto
     26140
26130 if isym=12 then pl$="lt"+str$(l%)+",0.5;"
26135 if isym=13 then pl$="lt"+str$(l%)+",1;"
26140 if isym>10 then gosub 25800
26145 return
26150 l%=dislin%(psym(j)-10) : call linetype(l%) :
     return
rem *******get characterset from system file *********
26400 close#2 : open "r",#2,name$+".HSY",256 :
     field#2,256 as ex$
26410 pointer%(25)=i : for i1=1 to 10 : i2=i1+(i-1)*10
     : get#2,i2
26430  il=cvi(mid$(ex$,1,2))
26440  sym$(i1)=mid$(ex$,3,il)
26450 next i1 : close#2 : ex$="" : return
rem **********interpolate Linie************************
26500 if psym(j)<11 then return
26520 if dmax(j)>250 then return
26530 rem
26550 for i=1 to dmax(j)+1 : rex(i)=dx(j,i) :
     rey(i)=dy(j,i) : next i
26560 rey(3)=rey(4)*2-rey(5)
26570 rey(dmax(j)+1)=rey(dmax(j))*2-rey(dmax(j)-1)
26580 j1=2 : deltay=abs(pyl#(2)-pyl#(1))/100 : ilast=0
26600 for i=4 to dmax(j) : del=abs(rey(i+1)-rey(i)) :
     if del>deltay/4 or ilast=0 or i=dmax(j) then dfl=1
     else dfl=0
26605 if del>deltay then dfl=2
26607 if (320-j1)<(dmax(j)-i) then if dfl=2 then dfl=1
```

```
26610 p=rey(i) : pp1=rey(i+1) : pm1=rey(i-1) : x=rex(i)
     : xp1=rex(i+1) : xm1=rex(i-1) : if xp1=xm1 then
     goto 26700
26620 h1=(pp1-pm1)/(xp1-xm1) : h2=((pp1+pm1-2*p)/(xp1-
     xm1))/2
26640 if dfl=2 then j1=j1+1 : dy(j,j1)=p+(h2-h1)*(x-
     xm1)/3 : dx(j,j1)=x-(x-xm1)/3
26650 if dfl>0 then j1=j1+1 : dy(j,j1)=p : dx(j,j1)=x :
     ilast=1 else ilast=0
26660 if dfl=2 then j1=j1+1 : dy(j,j1)=p+(h2+h1)*(xp1-
     x)/3 : dx(j,j1)=x+(xp1-x)/3
26700 next i : dmax(j)=j1-1
26790 return
rem ***************Columns Drawing ********************
26800 x2=(pxu#(2)-pxu#(1))/200 : y2=(pyl#(2)-
     pyl#(1))/200 : y0=pyl#(1)
26810 h1=x1-x2*pga(8) : h2=y0 : ipen=0 : gosub 26000 :
     xul%=xh% : yul%=yh% : h2=y1 : ipen=1 : gosub 26000
     : h1=x1+x2*pga(8) : gosub 26000 : ixor=xh% :
     yor%=yh%
26815 h2=y0 : gosub 26000 : h1=x1-x2*pga(8) : gosub
26000 : ipen=0 : gosub 26000
26820 if isym=21 then goto 26890
26825 if isym=22 then gosub 26900 : goto 26890
26830 if isym=23 then gosub 26920 : goto 26890
26835 if isym=24 then gosub 26900 : gosub 26920 : goto
     26890
26840 if isym=25 then goto 26890
26845 if isym=26 then goto 26890
26850 if isym=27 then goto 26890
26855 if isym=28 then goto 26890
26860 if isym=29 then pl$="lt"+str$(1%)+",0.5;" : gosub
25800 : gosub 26900 : goto 26890
26865 if isym=30 then gosub 26940 : goto 26890
```

```
26890 return
26900 xstart=x1-x2*pga(8) : xstop=x1+x2*pga(8)
26901 for schraffur=xstart to xstop step x2 :
     h1=schraffur : h2=y0 : ipen=0 : gosub 26000 :
     h2=y1 : ipen=1 : gosub 26000 : ipen=0 : gosub
26000 : next schraffur
26905 return
26920 for schraffur=y0 to y1 step 2*y2
26922 h1=x1-pga(8)*x2 : h2=schraffur : ipen=0 : gosub
26000 : h1=x1+pga(8)*x2 : ipen=1 : gosub 26000 : ipen=0
     : gosub 26000 : next schraffur
26925 return
26940 xstart=x1-x2*pga(8) : xstop=x1+x2*pga(8) :
     x3=x2/4
26941 for schraffur=xstart to xstop step x3 :
     h1=schraffur : h2=y0 : ipen=0 : gosub 26000 :
     h2=y1 : ipen=1 : gosub 26000 : ipen=0 : gosub
26000 : next schraffur
26945 return
27000 rem ************autoscaling routine**************
27020 x11=-999999 : x00=999999 : y11=-999999 :
     y00=999999
27050 for i=1 to 10 : if dmax(i)<5 then goto 27090
27060 for j=4 to dmax(i)
27065 if x00>dx(i,j) then x00=dx(i,j)
27070 if y00>dy(i,j) then y00=dy(i,j)
27075 if x11<dx(i,j) then x11=dx(i,j)
27080 if y11<dy(i,j) then y11=dy(i,j)
27085 next j
27090 next i : on error goto 21900
27092 x0#=x00 : y0#=y00 : x1#=x11 : y1#=y11
27095 x0#=int(x0#*10000)/10000 : x1#=int(x1#*
     10000)/10000
```

```
27100 y0#=int(y0#*10000)/10000 : y1#=int(y1#*
     10000)/10000
27105 if x0#>100000 then x0#=100000
27110 if x1#>1000000 then x1#=1000000
27115 if y0#>100000 then y0#=100000
27120 if y1#>1000000 then y1#=1000000
27125 if x0#<-100000 then x0#=-100000
27130 if x1#<-1000000 then x1#=-1000000
27135 if y0#<-100000 then y0#=-100000
27140 if y1#<-1000000 then y1#=-1000000
27150 min3#=x0# : min2#=x0# : min#=x0# : max#=x1# :
     achse=1 : gosub 27250
27160 pxu#(1)=x0# : pxu#(2)=x1# : pxu#(3)=x2# :
     pxu#(4)=x3# : pxo#(1)=x0# : pxo#(2)=x1# :
     pxo#(3)=x2# : pxo#(4)=x3#
27170 min#=y0# : max#=y1# : achse=2 : gosub 27250
27180 pyl#(1)=x0# : pyl#(2)=x1# : pyl#(3)=x2# :
     pyl#(4)=x3# : pyr#(1)=x0# : pyr#(2)=x1# :
     pyr#(3)=x2# : pyr#(4)=x3#
27190 on error goto 0 : return
27250 rem
***************autoscale**************************
27255 if min#>max# then h1#=min# : min#=max# : max#=h1#
27260 range#=(max#-min#) : scf#=1 : if range#=0 then
     goto 500
27265 if range#<100 then goto 27270
27267 range#=range#/10 : scf#=scf#*10 :  if
     abs(range#)>1000000 or  abs(range#)<0.0000001 then
     return else goto 27265
27270 if range#>10 then goto 27275
27272 range#=range#*10 : scf#=scf#/10 : if
     abs(range#)>1000000 or  abs(range#)<0.0000001 then
     return else goto 27270
```

```
2727[illegible] range#=cint(range#)+1 [illegible] range2#=int(range#/10[illegible] [illegible]
     range#=range#+range2#
27276 if range2#<2 then range2#=2 : goto 27280
27277 if range2#<3 then range2#=2.5 : goto 27280
27278 if range2#<6 then range2#=5 else range2#=10
27280 range#=range#*scf# : del#=scf# :
     range2#=range2#*scf#
27281 if min#<0 then scf#=-1 else scf#=1
27282 min#=min#*scf# : if min#=0 then goto 27296
27285 if min#<100 then goto 27290
27287 min#=min#/10 : scf#=scf#*10 :  if
     abs(min#)>1000000 or  abs(min#)<0.0000001 then
     return else goto 27285
27290 if min#>10 then goto 27294
27292 min#=min#*10 : scf#=scf#/10 :  if
     abs(min#)>1000000 or abs(min#)<0.0000001 then
     return else goto 27290
27294 min#=int(min#)+2 : min2#=int(min#/10)*10 :
     min#=min#*scf# : min2#=min2#*scf#
27295 if pret$="vsp" and achse=1 then goto 27400
27296 if abs(min#)>range#/2 then goto 27300 else goto
27350
27300 if abs(min#)>100*range# then x0#=h1# : x1#=max# :
     x2#=range2# : x3#=abs(h1#) : goto 27340
27310 if abs(min#)>10*range# then x0#=min# :
     x1#=min#+range# : x2#=range2# : x3#=abs(min#-
     min2#) : goto 27340
27320 if min#>0 then x0#=min2# : x1#=min2#+range# :
     x2#=range2# : x3#=0 : goto 27340
27330 if min#<0 then x0#=min#-scf# : x1#=x0#+range# :
     x2#=range2# : x3#=abs(min2#-x0#)
27340 return
```

```
27350 if min#>=0 then x0#=0 : x1#=range# : x2#=range2#
     : x3#=0 else x0#=min# : x1#=range#+min# :
     x2#=range2# : x3#=abs(min#)
27390 return
27400 x0#=min3# : x1#=max# : range#=max#-min3#
27410 if range#>400 then x2#=100 : x3#=int(min3#
     /100+1)*100-min3# : goto 27425
27415 if range#>100 then x2#=50 : x3#=int(min3
     #/50+1)*50-min3# : goto 27425
27420 if range#>20 then x2#=10 : x3#=int(min3#/
     10+1)*10-min3# else x2#=2 : x3#=int(min3#/2+1)*2-
     min3#
27425 return
27500 rem ***********  smooth  ***********************
27510 if dmax(j)<12 then goto 27760
27515 if dy(j,3)=0 then dy(j,3)=2*dy(j,4)-dy(j,5)
27520 dy(j,2)=2*dy(j,3)-dy(j,4) : dy(j,1)=2*dy(j,3)-
     dy(j,5) : j2=dmax(j)
27525 if dy(j,dmax(j)+1)=0 then dy(j,j2+1)=2*dy(j,j2)-
     dy(j,j2-1)
27530 dy(j,j2+2)=2*dy(j,j2+1)-dy(j,j2) : j1=1
27540 dy(j,j2+3)=2*dy(j,j2+1)-dy(j,j2-1) : on smooth
     goto 27550,27600,27650
27550 for i=j1+3 to j2 : dy(j,i-3)=(dy(j,i-
     1)+dy(j,i)+dy(j,i+1))/3 : next i : goto 27740
27600 dd1=dy(j,j1+2)-dy(j,j1+1) : dd2=dy(j,j1+3)-
     dy(j,j1+2) : dd3=dy(j,j1+4)-dy(j,j1+3)
27605 de1=dd2-dd1 : de2=dd3-dd2 : df1=de2-de1
27610 for i=j1+3 to j2 : dd4=dy(j,i+2)-dy(j,i+1) :
     de3=dd4-dd3 : df2=de3-de2 : dg=df2-df1
27620 dy(j,i-3)=dy(j,i)-3*dg/35 : dd1=dd2 : dd2=dd3 :
     dd3=dd4 : de1=de2 : de2=de3 : df1=df2
27630 next i : goto 27740
27650 dd1=dy(j,j1) : dd2=dy(j,j1+1) : dd3=dy(j,j1+2)
```

```
27655 for i=j1+3 to j2 : dd4=dy(j,i) : dy(j,i-3)=(-
    2*dd1+3*dd2+6*dd3+7*dd4+6*dy(j,i+1)+3*dy(j,i+2)-
    2*dy(j,i+3))/21
27665 dd1=dd2 : dd2=dd3 : dd3=dd4 : next i
27740 for i=dmax(j)-3 to 1 step-1 : dy(j,i+3)=dy(j,i) :
    next i
27750 rem
27760 return
rem *********************  cursor *******************
28000 if cursor=11 then goto 28060
28004 if cursor=12 then goto 28080
28010 if cw>dmax(cursor) then cw=dmax(cursor)
28012 if cw<3 then cw=3
28013 if omit%=1 then 28030
28015 gosub 22390 : gosub 22350 : print using "###
    ####.####  ####.####";cursor,dx(cursor,cw),
    dy(cursor,cw) : print chr$(27)+"*dS";
28030 xdc%=fnxw(dx(cursor,cw)) :
    ydc%=fnyw(dy(cursor,cw)) : if xdc%>511 then
    xdc%=511
28042 if ydc%>389 then ydc%=389
28044 if xdc%<1 then xdc%=1
28046 if ydc%<1 then ydc%=1
28048 call mgcurs(xdc%,ydc%) : return
28050 print fncurs$(dx(cursor,cw),dy(cursor,cw)) :
    return
28060 if x%>511 then x%=511
28062 if y%>389 then y%=389
28064 if x%<1 then x%=1
28066 if y%<1 then y%=1
28068 call mgcurs(x%,y%) : return
28080 h3=(x%-pxmin)/scxf#+scadx# : h4=(y%-
    pymin)/scyf#+scady# : gosub 22390
```

```
28082 gosub 22350 : print using "###  ####.#####
     ####.####";cursor,h3,h4
28086 return
rem **************Data and x-y cursor****************
28100 if a$="1" then cw=cw-1 : gosub 28140 : if ic>1
     then cw=cw-ic : gosub 28000 : goto 28100 else
     gosub 28000 : return
28110 if a$="5" then cursor=cursor+1 : if cursor>10
     then cursor=1 : gosub 28000 : return else gosub
     28000 : return
28120 if a$="2" then cursor=cursor-1 : if cursor<1 then
     cursor=10 : gosub 28000 : return else gosub 28000
     : return
28128 if a$="3" then cw=cw+1 : gosub 28140 : if ic>1
     then cw=cw+ic : gosub 28000 : goto 28128 else
     gosub 28000 : return
28135 return
28140 id=0 : ie=105-ic*10 : if ie<5 or ie>110 then ie=5
     : ic=10
28145 for i=1 to ie : b$=inkey$ : if len(b$)>0 then
     id=id+1
28150 next i : if id>0 then ic=ic+1 else ic=ic-3 : if
     ic<1 then ic=1
28155 if ic>3 then omit%=1 else omit%=0
28160 return
28200 if a$="0" then cursor=12 : gosub 28000 :
     cursor=11 : return
28208 if a$="1" then x%=x%-1 : gosub 28140 : if ic>1
     then x%=x%-ic : gosub 28000 : goto 28208 else
     gosub 28000
28209 if a$="5" then y%=y%+1 : gosub 28140 : if ic>1
     then y%=y%+ic : gosub 28000 : goto 28209 else
     gosub 28000
```

```
28210 if a$="2" then y%=y%-1 : gosub 28140 : if ic>1
     then y%=y%-ic : gosub 28000 : goto 28210 else
     gosub 28000
28211 if a$="3" then x%=x%+1 : gosub 28140 : if ic>1
     then x%=x%+ic : gosub 28000 : goto 28211 else
     gosub 28000
28212 return
rem ****************call size************************
29100 s%=int(si/pg(1)*(pxmax-pxmin)/4.5*256)
29103 if s%>2560 then s%=2560
29105 if s%<64 then s%=64
29110 sy%=int(siy/pg(2)*(pymax-pymin)/6.3*256)
29112 if sy%>2560 then sy%=2560
29113 if sy%<128 then sy%=sy%*2 : goto 29115
29114 if sy%<192 then sy%=sy%*1.5
29115 if sy%<64 then sy%=64
29122 call size(s%,sy%)
29125 si=s%/256*7 : siy=sy%/256*10 : return : rem size
     of character cell
29500 rem *********Defaults for positions  ***********
29510 on pg(13) goto 29520,29600,29700,29750
29520 pg(3)=2 : pg(4)=2 : return
29600 pg(4)=3 : if pg(9)=1 then pg(3)=1.2 else
     pg(3)=14.2
29660 return
29700 on pg(9) goto 29710,29715,29720,29725
29710 pg(3)=2 : pg(4)=11 : return
29715 pg(3)=15 : pg(4)=11 : return
29720 pg(3)=2 : pg(4)=2 : return
29725 pg(3)=15 : pg(4)=2 : return
29750 on pg(9) goto 29760,29765,29770,29775,29780,29785
29760 pg(3)=2 : pg(4)=2 : return
29765 pg(3)=2 : pg(4)=11 : return
29770 pg(3)=10 : pg(4)=2 : return
```

```
29775 pg(3)=10 : pg(4)=11 : return
29780 pg(3)=18 : pg(4)=2 : return
29785 pg(3)=18 : pg(4)=11 : return
29800 rem ***********Defaults for layouts  ************
29810 on pg(13) goto 29820,29840,29860,29880
29820 pg(1)=20 : pg(2)=15
29825 pgx(1)=.3 : pgx(2)=.25 : pgx(3)=.25 : pgx(4)=.2 :
     pgx(5)=.2
29830 pgy(1)=.4 : pgy(2)=.35 : pgy(3)=.3 : pgy(4)=.25 :
     pgy(5)=.25 : return
29840 pg(1)=15 : pg(2)=10
29845 pgx(1)=.25 : pgx(2)=.20 : pgx(3)=.20 : pgx(4)=.15
     : pgx(5)=.15
29850 pgy(1)=.3 : pgy(2)=.30 : pgy(3)=.25 : pgy(4)=.20
     : pgy(5)=.25 : return
29860 pg(1)=10 : pg(2)=7
29865 pgx(1)=.16 : pgx(2)=.15 : pgx(3)=.14 : pgx(4)=.12
     : pgx(5)=.12 : pgx(8)=-12 : pgy(7)=-12
29870 pgy(1)=.22 : pgy(2)=.19 : pgy(3)=.17 : pgy(4)=.15
     : pgy(5)=.19 : return
29880 pg(1)=7 : pg(2)=5
29885 pgx(1)=.16 : pgx(2)=.15 : pgx(3)=.14 : pgx(4)=.12
     : pgx(5)=.12 : pgx(8)=-12 : pgy(7)=-12
29890 pgy(1)=.22 : pgy(2)=.19 : pgy(3)=.17 : pgy(4)=.15
     : pgy(5)=.19 : return
rem ****************Setup Touchsreen ******************
30000 on error goto 0 : rowinc%=0
30005 row%=1 : col%=20 : colinc%=59: resp$="T" : gosub
     30200
30010 row%=3 : col%=20 : colinc%=15: resp$="A" : gosub
     30200
30015 row%=3 : col%=68 : resp$="Q" : gosub 30200
30020 row%=5 : col%=20 : resp$="M" : gosub 30200
30025 row%=5 : col%=68 : resp$="N" : gosub 30200
```

```
30030 row%=8 : col%=0 : colinc%=10 : resp$="G" : gosub
     30200
30035 row%=10 : resp$="X" : gosub 30200 : row%=12 :
     resp$="Y" : gosub 30200
30040 row%=14 : resp$="V" : gosub 30200 : row%=16 :
     resp$="W" : gosub 30200
30045 row%=18 : resp$="B" : gosub 30200 : row%=20 :
     resp$="Y" : gosub 30200 : colinc%=15
30050 row%=26 : resp$="E" : gosub 30200 : row%=28 :
     resp$="F" : gosub 30200
30055 row%=30 : resp$="G" : gosub 30200 : row%=32 :
     resp$="H" : gosub 30200
30060 for j=8 to 20 step 2 : row%=j
30062 for i=1 to 8 : col%=i*8+4 : if i=5 then
     colinc%=35 else colinc%=6
30064 if j>17 then if i>5 then goto 30072 else
     colinc%=6
30066 if j=8 then colinc%=6 : goto 30070
30068 if i>5 then goto 30072
30070 resp$=chr$((j-8)*4+i+89) : gosub 30200
30072 next i : next j : colinc%=8
30080 for j=26 to 32 step 2 : row%=j
30082 for i=1 to 6 : col%=i*10+8 : resp$=chr$((j-
     26)*3+i+145) : gosub 30200 : next i
30084 next j
30090 for j=35 to 41 step 2 : row%=j
30092 for i=1 to 4 : col%=(i-1)*8 : if i=4 then
     colinc%=52 else colinc%=6
30094 resp$=chr$((j-35)*2+i+169) : gosub 30200
30096 next i : next j
30190 return
rem ***************call fntouch**********************
30200 call fntouch(row%,col%,rowinc%,colinc%,resp$)
30210 return
```

```
rem **********input routine*****************************
40000 icol%=col%+len(prompt$) : il=len(in$) : if
     il>ileer or icol%>79 then in$=space$(ileer) :
     return
40020 if icol%+ileer>80 then ileer=80-icol%
40030 if il<ileer then in$=in$+space$(ileer-il)
40040 call curs(col%,row%) : print prompt$;in$;"*" :
     call curs(icol%,row%) : ia=0 : tshare=4
40050 while ia<>13 : call echo(ia) : if ia>17 and ia<27
     or ia=189 then 40190
40060 iplot=iplot+1 : if iplot>tshare then iplot=0 : if
     pointer%(1)>0 then gosub 4300
40080 wend
40100 h1$=space$(80)+"" : call readli(row%,h1$)
40110 for ii=1 to ileer : if mid$(h1$,icol%+ii,ileer-
     ii+1)=space$(ileer-ii+1) then 40150
40120 next ii : ii=ileer
40150 in$=mid$(h1$,icol%+1,ii)
40190 h1$="" : return

rem  ***************************************************
rem  ******    MICHFIT software :  dacom.bas    ********
rem  ******    Author  :  Bruno Michel          *********
rem  ***************************************************
rem
rem
rem ****************************************************
rem
rem
rem *********************************************
10 rem $include : 'comvar'
220 width 255

rem *********************************************
```

```
41000 call cls : print "program dacom B.Michel 02. Feb.
     88    e=exit" : on error goto 42000
41020 defint i : ioshare=255 : iopen=0 : ihandle=0
41030 ix=0 : call popen(ix) : ihandle=ix
41040 rsio$=""
41050 for inlop=1 to ioshare : a$=inkey$ : if a$="e"
     then 41190
41060  i1=ihandle : call instat(i1) : if i1<>255 then
     41040
41070  ix=ihandle : call rsin(ix) : if ix=10 then gosub
41200 : goto 41100
41080  if ix<>13 then rsio$=rsio$+chr$(ix)
41090 next inlop : goto 41120
41100 print rsio$
41120 goto 41040 : rem *******return*****************
41190 ix=ihandle : call pclose(ix) : forts=1 : pinit=1
     : chain "ha"
rem *************command interpreter*************
rem
41200 if len(rsio$)<5 then 41230
41210 if mid$(rsio$,1,5)="OPEN " then if len(rsio$)>19
     then gosub 41400 : return
41220 if mid$(rsio$,1,5)="CLOSE" then close#15 :
     iopen=0 : return
41230 if mid$(rsio$,1,5)="HELLO" then gosub 41300 :
     return
41250 if iopen=1 then print#15,rsio$
41260 return
rem ***********setup datacommunication ***************
41300 rsio$="hello" : i1=5 : print "quittiere HELLO"
41310 for i=1 to i1+2 : a1$=inkey$ : if a1$="e" then
     41380
41320   i2=ihandle : call outstat(i2) : if i2<>255 then
     return
```

```
41330   if i=il+1 then ii=13
41340   if i=il+2 then ii=10
41350   if i<=il then ii=asc(mid$(rsio$,i,1))
41360   ix=ihandle : call rsout(ix,ii)
41370 next i : print "ok hello"
41380 return
rem ****************open file *************************
rem ************and store data *************************
41400 h1$=mid$(rsio$,6,14) : iopen=1 : close#15 : open
     "o",#15,h1$
41405 rsio$="file is open" : il=12
41410 for i=1 to il+2 : a1$=inkey$ : if a1$="e" then
     41380
41420   i2=ihandle : call outstat(i2) : if i2<>255 then
     return
41430   if i=il+1 then ii=13
41440   if i=il+2 then ii=10
41450   if i<=il then ii=asc(mid$(rsio$,i,1))
41460   ix=ihandle : call rsout(ix,ii)
41470 next i
41480 return




42000 resume 41020

rem  **************************************************
rem  *****     MICHFIT software :  comvar.bas        ***
rem  ******     Author : Bruno Michel          *********
rem  **************************************************
rem
rem
rem
```

```
rem
rem
rem ****************************************************
rem              definition of global variables
rem ****************************************************
rem
rem
rem option base 1   start counting from 1 for array
     numbering
rem defint i,j,w,t  all variables starting with i,j,w,t
     are
rem              integer
rem dim format$(90),
rem  sym$(10),
rem  sys$(10),
rem  inhalt$(10),
rem  beschr$(20),
rem  macro$(20),
rem  filename$(20)
rem dim icon(450),
rem  idrive(20)
rem dim psym(10),
rem  dmax(10),
rem  stift%(10),
rem  tl(30),
rem  tque(10),
rem  sque(20),
rem  inhp(10),
rem  c(10)
rem dim dx(10,332),
rem  dy(10,332),
rem  pg(30),
rem  pxu#(5),
rem  pyl#(5),
rem  pxo#(5),
rem  pyr#(5),
rem  pgx(30),
rem  pgy(30),
rem  pga(30)
rem dim ivar(10,100),
rem  nvar(10,100),
rem  idat(10),
rem  idx(128),
rem  idy(128),
rem  idp(128),
rem  pointer%(30),
rem  iu(100),
rem  ip(100)
rem
rem
rem
rem ****************************************************
rem                common variables
rem ****************************************************
rem
rem
rem
rem common pinit,
rem  forts,
rem  tmod,
rem  taus,
rem  tein,teart,idat()
rem common name$,
rem  initialen$,
rem  pret$,
rem  datst$
rem common anzgem,
rem  anztit,
rem  anzspek,
rem  autoscale,
rem  interpolate,
rem  anzeige,
rem  default
rem common softk,
rem  anzart,
rem  offo%,
rem  offi%
rem common format$(),
ren  sym$(),
rem  sys$(),
rem  inhalt$(),
rem  beschr$(),
rem  macro$(),
rem  filename$()
rem common icon(),
rem  idrive()
rem common psym(),
rem  dmax(),
rem  stift%(),
rem  tl(),
rem  tque(),
rem  sque(),
rem  inhp(),
rem  c(10)
rem common dx(),
rem  dy(),
rem  pg(),
rem  pxu#(),
rem  pyl#(),
rem  pxo#(),
rem  pyr#(),
rem  pgx(),
rem  pgy(),
rem  pga()
rem common ivar(),
rem  nvar(),
rem  idx(),
rem  idy(),
rem  idp(),
rem  pointer%(),
rem  iu(),
rem  ip()
rem
rem
rem ***************************************************
rem                  start of source code
rem ***************************************************
rem
rem
rem *********** declaration of variables ************
100 option base 1
105 defint i,j,w,t
110 dim format$(90),sym$(10),sys$(10),inhalt$(10),
     beschr$(20),macro$(20),filename$(20)
112 dim icon(450),idrive(20)
115 dim psym(10),dmax(10),stift%(10),tl(30),tque(10),
     sque(20),inhp(10),c(10)
120 dim dx(10,332),dy(10,332),pg(30),pxu#(5),pyl#(5),
     pxo#(5),pyr#(5),pgx(30),pgy(30),pga(30)
125 dim ivar(10,100),nvar(10,100),idat(10),idx(128),
     idy(128),idp(128),pointer%(30),iu(100),ip(100)
rem
rem **************** common declarations *************
rem
140 common pinit,forts,tmod,taus,tein,teart,idat()
145 common name$,initialen$,pret$,datst$
```

```
150 common anzgem,anztit,anzspek,autoscale,interpolate,
     anzeige,default
152 common softk,anzart,offo%,offi%

160 common format$(),sym$(),sys$(),inhalt$(),beschr$(),
     macro$(),filename$()
165 common icon(),idrive()
170 common psym(),dmax(),stift%(),tl(),tque(),sque(),
     inhp(),c(10)
175 common dx(),dy(),pg(),pxu#(),pyl#(),pxo#(),pyr#(),
     pgx(),pgy(),pga()
180 common ivar(),nvar(),idx(),idy(),idp(),pointer%(),
     iu(),ip()
rem
rem
rem ****************************************************
```

APPENDIX F
System Software Listings

```
; ****************************************************
; ****    MICHFIT software :  dosi.asm      **********
; ************    Author :   Bruno Michel    **********
; ****************************************************
;
;
;
;
; ****************************************************
;    Assembly language program for the implementation
;    of interrupts to the MICROSOFT DOS operating
;    system vesion 2.11.
;    In addition to that it contains a routine that
;    implements virtual alpha-numeric screens.
; ****************************************************
;
; The file DC.EQU containing several macros is included
; when this progaram is assembled.

INCLUDE   DC.EQU
;
;
; The following macro is used to read one line from the
; alphanumeric screen and to store it into a buffer.

SCHREIB   MACRO      DEFSCHR,SCHRBUF
          MOV        AX,4403H
          MOV        BX,1
          MOV        CX,10
          MOV        DX,OFFSET DEFSCHR
          INT        21H
```

```
                    MOV       AX,4403H
                    MOV       BX,1
                    MOV       CX,16
                    MOV       DX,OFFSET SCHRBUF
                    INT       21H
                    ENDM


; The following macro is used to read one line from a
; buffer and to write it to the alphanumeric screen.

LESELI              MACRO     DEFLESE,LESEBUF
                    MOV       AX,4403H
                    MOV       BX,1
                    MOV       CX,10
                    MOV       DX,OFFSET DEFLESE
                    INT       21H
                    MOV       AX,4403H
                    MOV       BX,1
                    MOV       CX,16
                    MOV       DX,OFFSET LESEBUF
                    INT       21H
                    ENDM

; All code and data aof the assembly language calls is
; linked to the BASIC code segment.

CODE SEGMENT        PARA PUBLIC    'CODE'
ASSUME              CS:CODE,DS:CODE,ES:CODE

; The following public declarations are recognized by
; the linker so that the routines can be acessed from
; the outside of the object code.
```

```
        PUBLIC      SUCHF,SUCHN,DATUM,ZEIT,WART,DISLI,READL,
                    ECHO,STOSCR,RECSCR
        PUBLIC      POPEN,INSTAT,RSIN,OUTSTAT,RSOUT,PCLOSE


; *****************************************************
; The following function is used to search through the
; sub directory or the disc drive passed to the routine
; in the pfad$ string. If a has been found the name of
; the file is returned in the filef$ string. If no file
; has been found or if the drive or subdirectory does
; not exist, an error code is  returned in the segment%
; integer variable. The strings are passed to the
; assembly language routine by means of pointers to
; sting descriptors.
; Because the actual data of the strings resides in the
; basic data area and only an offset to the BASIC's
; data segment is stored, the address of the  BASIC
; data segment has to be passed too.

PFADS       EQU         10
PFAD        EQU         8
FILEF       EQU         6
SUCHST      EQU         3*2
SUCHF       PROC        FAR         ; syntax : SUCHF(segment
                                          %,pfad$,filef$)
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            PUSH        ES
            PUSH        DS
; copy path string from BASIC data area to the buffer
; PFADNAME
            MOV         BX,PFAD[BP]
```

```
          MOV       CX,[BX]                ;LENGTH OF
                                           PFAD$ ID CX
          MOV       SI,2[BX]               ;ADDRESS OF
                                           PFAD$ IN DX
          MOV       AX,SEG BUFFER
          MOV       ES,AX
          MOV       DI,OFFSET PFADNAME

COPYF:    MOV       AL,BYTE PTR[SI]
          MOV       ES:BYTE PTR[DI],AL
          INC       SI
          INC       DI
          LOOP      COPYF
          MOV       AL,0H                  ;Prepare
                                           processor
                                           registers for
          MOV   ES:BYTE PTR[DI],AL         ;interrupt to
                                           DOS
          MOV       AX,SEG CODE
          MOV       DS,AX
          MOV       ES,AX
          SUB       AX,AX
          MOV       DX,OFFSET BUFFER
          MOV       AH,1AH
          INT       21H               ;Execute interrupt
          MOV       DX,OFFSET PFADNAME
          MOV       CX,0
          MOV       AH,4EH
          INT       21H
          POP       ES
          CMP       AL,18                  ;Has a file
                                           been found?
          JE        NEIN
          MOV       BX,FILEF[BP]
          MOV       DI,2[BX]
          MOV       CX,12
          MOV       AX,SEG BUFFER
          MOV       DS,AX
          MOV       SI,OFFSET BUFFER
          ADD       SI,1EH
CPFF:                         ;If yes  copy filename to
                              BASIC data area.
          MOV       AL,BYTE PTR[SI]
          MOV       ES:BYTE PTR[DI],AL
          INC       SI
          INC       DI
          LOOP      CPFF
          MOV       AL,0
NEIN:                         ;return flag in segment%
          MOV       AH,0
          MOV       BX,PFADS[BP]
          MOV       [BX],AX
          POP       ES
          POP       DS
          POP       BP
          RET       SUCHST
SUCHF     ENDP

; *****************************************************
; If a file has been found by the SUCHF routine this
; routine can be used to  step through the directory
; for more files until the last one has been found.
; In this case, the flag is set.


FILENS    EQU       8
FILEN     EQU       6
SUCHSTN   EQU       2*2
```

```
SUCHN    PROC     FAR        ;Syntax:
                             SUCHN(segment%,file
                             name$)
         PUSH     BP
         MOV      BP,SP
         PUSH     DS
         PUSH     ES
         PUSH     DS
         PUSH     DS

         SUB      AX,AX
         MOV      AH,4FH
         INT      21H              ;Execute interrupt
         POP      ES
         CMP      AL,18            ;Has a file been
                                   found?
         JE       ENDE
         MOV      BX,FILEN[BP]
         MOV      DI,2[BX]
         PUSH     DI
         MOV      CX,8
         MOV      AX,SEG BUFFER
         MOV      DS,AX
         MOV      SI,OFFSET BUFFER
         ADD      SI,1EH
CPFN:                              ;If yes copy
                                   filename to BASIC
                                   data area
         MOV      AL,BYTE PTR[SI]
         CMP      AL,2EH
         JE       CPEXT
         CMP      AL,0
```

```
                JE          NOEXT
                MOV         ES:BYTE PTR[DI],AL
                INC         SI
                INC         DI
                LOOP        CPFN
        CPEXT:                                  ;copy extension
                POP         DI
                ADD         DI,9
                MOV         CX,3
                INC         SI
        EXTL:
                MOV         AL,BYTE PTR[SI]
                MOV         ES:BYTE PTR[DI],AL
                INC         SI
                INC         DI
                LOOP        EXTL
                MOV         AL,0
        ENDE:                                   ;load flag
                MOV         AH,0
                POP         DS
                MOV         BX,FILENS[BP]
                MOV         [BX],AX
                POP         ES
                POP         DS
                POP         BP
                RET         SUCHSTN
        NOEXT:
                MOV         AL,0
                POP         DI
                JMP         ENDE
        SUCHN       ENDP

        ; *****************************************************
```

```
; The following function reads the internal clock and
; passes three integers to the BASIC application : the
; year, the day and the day of the week. From the TAG
; variable the month can be calculated  : month=TAG/256
; and day of month=
; TAG MOD 256. From this information a string is
; generated :
; Tuesday 02. Feb. 1988

JAHR        EQU         10
TAG         EQU         8
WTAG        EQU         6
DATST       EQU         3*2

DATUM       PROC        FAR         ;Syntax : DATUM
                                    (jahr%,tag%,wtag%)
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            GET_DATE
            MOV         BX,JAHR[BP]
            MOV         [BX],CX
            MOV         BX,TAG[BP]
            MOV         [BX],DX
            MOV         BX,WTAG[BP]
            MOV         [BX],AX
            POP         DS
            POP         BP
            RET         DATST
DATUM     ENDP

; ******************************************************
; The following function reads the internal clock and
; passes two integers to; the application. From the MIN
; variable the hours and minutes of the day can; be
; calculated : hours=MIN/256  and minutes=MIN mod 256.
; From the SEK variable; the  seconds and milliseconds
; can be calculated : seconds=sek/256 and;
; milliseconds=(SEK mod 256)*1000/256

MIN         EQU         8
SEK         EQU         6
ZEITST      EQU         2*2

ZEIT        PROC        FAR         ;Syntax : ZEIT(min%,sek%)
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            GET_TIME
            MOV         BX,MIN[BP]
            MOV         [BX],CX
            MOV         BX,SEK[BP]
            MOV         [BX],DX
            POP         DS
            POP         BP
            RET         ZEITST
ZEIT        ENDP

; *****************************************************
; The following routine waits for n times 4
; milliseconds. The n is passed to; the routine in the
; DAUER integer by means of a stack.

DAUER       EQU         6
DAUST       EQU         1*2

WART        PROC        FAR         ;Syntax : WART(dauer%)
            PUSH        BP
```

```
          MOV       BP,SP
          PUSH      DS
          MOV       AX,DAUER[BP]
BACK:     NOP
          MOV       CX,255
BACK1:    NOP
          LOOP      BACK1
          DEC       AX
          CMP       AX,0
          JA        BACK
          POP       DS
          POP       BP
          RET       DAUST
WART      ENDP


; *****************************************************
; The following routine displays a line of text stored
; in the STRI string on; the line LINE of the
; alphanumeric screen with the enhancement stored in
; the  string ENHA. The strings are passed to the
; assembly     language routine by means of pointers to
; string          descriptors which in turn contain the
; offset of the     first byte to the BASIC data
; segment and the string     length in bytes.
; The line integer can range from 0 to 47.

LINIE     EQU       10
STRI      EQU       8
ENHA      EQU       6
POPLI     EQU       3*2

DISLI     PROC      FAR       ;Syntax DISLI(line%,
                              stri$,enha$)
```

```
                PUSH    BP
                MOV     BP,SP
                PUSH    DS
                MOV     BX,STRI[BP]
                MOV     CX,[BX]             ;LENGTH OF
                                            STRI$ ID CX
                MOV     SI,2[BX]            ;ADDRESS OF
                                            STRI$ IN DX
;

                MOV     AX,SEG CODE
                MOV     ES,AX
                MOV     DI,OFFSET CHAR_PTR
                MOV     BX,LINIE[BP]
                MOV     AL,[BX]
                MOV     ES:ALIN1,AL
                MOV     ES:ALIN2,AL

COPYS:          MOV     AL,BYTE PTR[SI]     ;copy string to
                                            command buffer
                MOV     ES:BYTE PTR[DI],AL
                INC     SI
                INC     DI
                LOOP    COPYS

                MOV     BX,ENHA[BP]
                MOV     CX,[BX]         ;LENGTH OF ENHA$ ID
                                        CX
                MOV     SI,2[BX]        ;ADDRESS OF ENHA$ IN
                                        DX
                MOV     DI,OFFSET EMH_PTR

COPYE:          MOV     AL,BYTE PTR[SI]     ;copy
                                            enhancement to
                                            command buffer
```

```
          MOV       ES:BYTE PTR[DI],AL
          INC       SI
          INC       DI
          LOOP      COPYE
;set up processor registers for block write interrupt
          MOV       AX,SEG CODE
          MOV       DS,AX
          MOV       SEG1,AX
          MOV       SEG2,AX
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,10
          MOV       DX,OFFSET DN_BUF
          INT       21H           ;execute interrupt
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,16
          MOV       DX,OFFSET WN_BUF
          INT       21H

          POP       DS
          POP       BP
          RET       POPLI

DISLI     ENDP

; *****************************************************
; The following routine stores the line RLINE from the
; alphanumeric screen to  the STRI variable in the data
; segment of BASIC. The string is copied to an existing
; string and care has to be taken that the end line
; mark is not over- written.

RLINE     EQU       8
```

```
RSTR      EQU     6
READST    EQU     2*2

READLI    PROC    FAR          ;Syntax READLI
                               (rline%,rstr$)
          PUSH    BP
          MOV     BP,SP
          PUSH    DS
          PUSH    DS
          MOV     BX,RLINE[BP]    ;zeile 1-47
          MOV     AL,[BX]
          MOV     CS:RLIN1,AL
          MOV     CS:RLIN2,AL
          MOV     BX,RSTR[BP]
          MOV     CX,80           ;LENGTH OF STRI$ ID
                                  CX
          MOV     DI,2[BX]        ;ADDRESS OF STRI$ IN
                                  DX

          MOV     AX,SEG    CODE
          MOV     DS,AX
          MOV     ES,AX
          MOV     RSEG1,AX
          MOV     RSEG2,AX
          MOV     SI,OFFSET CHARBU
          PUSH    BX
          PUSH    CX
          PUSH    DI
          PUSH    SI
;set up processor registers for block copy interrupt
          MOV     AX,4403H
          MOV     BX,1
          MOV     CX,10
          MOV     DX,OFFSET DEFREAD
```

```
                    INT         21H
                    MOV         AX,4403H
                    MOV         BX,1
                    MOV         CX,16
                    MOV         DX,OFFSET READBU
                    INT         21H

                    POP         SI
                    POP         DI
                    POP         CX
                    POP         BX
          ASSUME    ES:NOTHING
                    POP         ES
          COPYR:    MOV         AL,DS:BYTE PTR[SI]   ;copy 80 bytes
                                                     to BASIC data
                                                     area
                    CM          PAL,0DH
                    JE          ENDCP
                    MOV         ES:BYTE PTR[DI],AL
                    INC         SI
                    INC         DI
                    LOOP        COPYR
          ENDCP:
                    POP         DS
                    POP         BP
                    RET         READST

          READLI    ENDP


          ; ******************************************************
          ; The following routine accepts inputs from the
          ; keyboard and displays the  characters on the terminal
          ; at the current cursor location. In addition to that
```

```
; the character is passed to the application in the
; char integer (0-255)

CHAR      EQU     6
CHARST    EQU     1*2

ECHO      PROC    FAR       ;Syntax : ECHO (char%)
          PUSH    BP
          MOV     BP,SP
          PUSH    DS
          MOV     AL,1
          MOV     AH,0CH
          INT     21H
          MOV     AH,0
          POP     DS
          MOV     BX,CHAR[BP]
          MOV     [BX],AX
          POP     BP
          RET     CHARST
ECHO ENDP


; *****************************************************
; The following routine copies the entire alphanumeric
; screen to a BASIC IO buffer. From there it can be
; written to disc into a random access file containing
; at least 20 screens.


OBUF      EQU     6
STOALPST  EQU     1*2

STOSCR    PROC    FAR       ;Syntax : STOSCR (obuf$)
          PUSH    BP
          MOV       BP,SP
          PUSH      DS
          PUSH      ES
          PUSH      DS
          POP       ES
          MOV       BX,OBUF[BP]
          MOV       DI,[BX]
          ADD       DI,188

          MOV       AX,SEG CODE
          MOV       DS,AX
          MOV       RSEG1,AX
          MOV       RSEG2,AX
          MOV       BL,0
          MOV       CX,22          ;copy 22 lines from
                                   alphanumeric memory
LESELOP:
          PUSH      CX
;
          PUSH      BX
          PUSH      DI
          PUSH      ES
          MOV       AX,SEG CODE
          MOV       ES,AX
          MOV       RLIN1,BL
          MOV       RLIN2,BL
          LESELI    DEFREAD,READBU      ;execute macro
                                        (see top)
          MOV       SI,OFFSET CHARBU
ASSUME    ES:NOTHING
          POP       ES
          POP       DI
          POP       BX
          MOV       CX,80
```

```
;copy 80 bytes to BASIC IO buffer

LESCOP:
          MOV       AL,BYTE PTR[SI]
          MOV       ES:BYTE PTR[DI],AL
          INC       DI
          INC       SI
          LOOP      LESCOP
;
          INC       BL
          POP       CX
          LOOP      LESELOP

          POP       ES
          POP       DS
          POP       BP
          RET       STOALPST
STOSCR    ENDP
;


; *******************************************************
; The following routine copies the information stored
; in the BASIC IO buffer  to the alphanumeric screen.
; The IO buffer has to be filled prior to the
; execution of this routine with data from the xxx.KSC
; random acess file on the winchester disc.

IBUF      EQU       6
RECALPST  EQU       1*2

RECSCR    PROC      FAR       ;Syntax : STOSCR (ibuf$)
          PUSH      BP
          MOV       BP,SP
          PUSH      AX
          PUSH      BX
          PUSH      CX
          PUSH      DX
          PUSH      SI
          PUSH      DI
          PUSH      DS
          PUSH      ES
          PUSH      SS
          PUSH      DS
          POP       ES

          MOV       BX,IBUF[BP]
          MOV       SI,[BX]
          ADD       SI,188

          MOV       AX,SEG CODE
          MOV       DS,AX
          MOV       SEG1,AX
          MOV       SEG2,AX
          MOV       BL,0
          MOV       CX,22
SCHRLOP:
          PUSH      CX
;
          MOV       CX,80
          MOV       DI,OFFSET CHAR_PTR
SCHRC:
          MOV       AL,ES:BYTE PTR[SI]
          MOV       BYTE PTR[DI],AL
          INC       DI
          INC       SI
          LOOP      SCHRC
```

```
                  PUSH      SI
                  PUSH      ES
                  MOV       AX,SEG CODE
                  MOV       ES,AX
                  MOV       ALIN1,BL
                  MOV       ALIN2,BL
                  PUSH      BX
                  SCHREIB   DN_BUF,WN_BUF
                  POP       BX
                  POP       ES
                  POP       SI
                  INC       BL
                  POP       CX
                  LOOP      SCHRLOP

                  POP       SS
                  POP       ES
                  POP       DS
                  POP       DI
                  POP       SI
                  POP       DX
                  POP       CX
                  POP       BX
                  POP       AX
                  POP       BP
                  RET       RECALPST
RECSCR            ENDP
;

; ******************************************************
; IO buffer for directory search;
;
BUFFER    DB   128 DUP (?)
```

```
CRLF DB   10,13,"$"
PFADNAME  DB   32 DUP (?)
;
; ******************************************************
; IO buffer for store and read alphanumeric screen

DN_BUF          DW          1
                DB          79
ALIN1           DB          0
                DB          0
ALIN2           DB          0
                DW          0FFFFH
                DW          0FFFFH

WN_BUF          DW          2
                DW          80
                DW          EMH_PTR
SEG1            DW          ?
                DW          -1
                DW          -1
                DW          CHAR_PTR
SEG2            DW          ?
EMH_PTR         DB        '@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@'
CHAR_PTR  DB    '

DEFREAD         DW          1
                DB          79
RLIN1           DB          0
                DB          0
RLIN2           DB          0
                DW          0FFFFH
                DW          0FFFFH
```

```
READBU          DW          5
                DW          80
                DW          ENHBU
RSEG1           DW          ?
                DW          -1
                DW          -1
                DW          CHARBU
RSEG2           DW          ?
ENHBU           DB      '@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@'
CHARBU          DB          '
                DB          10,13,"$"

; ******************************************************
; IO routines (see FNIO)

HANDLE     EQU        6
OPENST     EQU        1*2

POPEN      PROC       FAR
           PUSH       BP
           MOV        BP,SP
           PUSH       DS
           MOV        AX,SEG CODE
           MOV        DS,AX
           MOV      AX,3D02H     ; OPEN PORT 1
           MOV        BX,0
           MOV        CX,0
           MOV        DX,OFFSET COMNAM
           INT        21H
           MOV        CS:HAND_1,AX    ; SAVE HANDLE RETURN
           MOV        BX,AX
           MOV        AX,4403H
           MOV        CX,2
```

```
          MOV       DX,OFFSET MOD_BUF
          INT       21H
;         MOV       AX,4403H
;         MOV       BX,CS:HAND_1
;         MOV       CX,2
;         MOV       DX,OFFSET TRANSPM
;         INT       21H
          MOV       AX,HAND_1
          POP       DS
          MOV       BX,HANDLE[BP]
          MOV       [BX],AX
          POP       BP
          RET       OPENST
POPEN   ENDP

;         MOV       AX,4400H
;         LEA       DX,CHG_RAW      ; SWITCH TO RAW MODE
;         XOR       CX,CX
;         MOV       BX,HAND_1
;         INT       21H
;         MOV       KP_PRT,DX
;         OR        DX,0020H   ; IO CONTROL
;         XOR       DH,DH
;         MOV       AX,4401H
;         INT       21H
;         MOV       AX,4403H
;         LEA       DX,CHG_TRN
;         MOV       CX,0002H
;         MOV       BX,HAND_1
;         INT       21H

STATI     EQU       6
INSTACK   EQU       1*2
```

```
INSTAT     PROC       FAR
           PUSH       BP
           MOV             BP,SP
           PUSH       DS
           MOV        SI,STATI[BP]
           MOV        BX,[SI]
           MOV        AX,SEG CODE
           MOV        DS,AX
           MOV        AX,4406H
           MOV        BX,CS:HAND_1
           INT        21H
           POP        DS
           MOV        BX,STATI[BP]
           MOV        [BX],AX
           POP        BP
      RET  INSTACK
INSTAT   ENDP

STATO      EQU        6
OUSTACK    EQU        1*2

OUTSTAT    PROC       FAR
           PUSH BP
           MOV        BP,SP
           PUSH       DS
           MOV        SI,STATO[BP]
           MOV        BX,[SI]
           MOV        AX,SEG CODE
           MOV        DS,AX
           MOV        AX,4407H
           MOV        BX,CS:HAND_1
           INT        21H
           POP        DS
           MOV        BX,STATO[BP]
```

```
                MOV       [BX],AX
                PO        PBP
                RET       OUSTACK
OUTSTAT ENDP

CLOSHA          EQU       6
CLOSST          EQU       1*2

PCLOSE          PROC      FAR
                PUSH      BP
                MOV       BP,SP
                PUSH      DS
                MOV       SI,CLOSHA[BP]
                MOV       BX,[SI]
                MOV       AX,SEG CODE
                MOV       DS,AX
                MOV       AX,3E00H
                MOV       BX,CS:HAND_1
                INT       21H
                POP       DS
                POP       BP
                RET       CLOSST
PCLOSE  ENDP

;               MOV       AX,4403H
;               LEA       DX,CHG_NRM
;               MOV       CX,0002H
;               MOV       BX,HAND_1
;               INT       21H
;               MOV       DX,KP_PRT
;               XOR       DH,DH
;               MOV       AX,4401H
;               XOR       CX,CX
;               MOV       BX,HAND_1
;           INT         21H
;           MOV         AX,3E00H
;           MOV         BX,HAND_1
;           INT         21H

RSCHAR      EQU         6
RSINST      EQU         1*2

RSIN PROC   FAR
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            MOV         SI,RSCHAR[BP]
            MOV         BX,[SI]
            MOV         AX,SEG CODE
            MOV         DS,AX
            MOV         AX,4402H
            MOV         BX,CS:HAND_1
            MOV         CX,1
            MOV         DX,OFFSET ZEICHEN
            INT         21H
            MOV         AH,0
            MOV         AL,ZEICHEN
;           MOV         AX,99
            POP         DS
            MOV         BX,RSCHAR[BP]
            MOV         [BX],AX
            POP         BP
            RET         RSINST
RSIN ENDP

RSOHAN      EQU         8
RSOUV       EQU         6
RSOUST      EQU         2*2
```

```
RSOUT       PROC        FAR
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            MOV         SI,RSOUV[BP]
            MOV         DX,[SI]
            MOV         SI,RSOHAN[BP]
            MOV         BX,[SI]
            MOV         AX,SEG CODE
            MOV         DS,AX
            MOV         ZEICHEN,DL
            MOV         DX,OFFSET ZEICHEN
            MOV         AX,4000H
            MOV         BX,CS:HAND_1
            MOV         CX,1
            INT         21H
            POP         DS
            POP         BP
            RET         RSOUST
RSOUT       ENDP

HAND_1      DW          ?
COMNAM      DB          'COM1',0
MOD_BUF     DB          2,8
TRANSPM     DB          3,8
ZEICHEN     DB          ?

CODE      ENDS
END

;   **************************************************
;   **********   MICHFIT software :  fnt.asm   ********
;   **********   Author :  Bruno Michel         ********
;  *****************************************************
;
;
;
; *****************************************************
;    Assembler program for the implementation of
; touchscreen and graphic functions
; *****************************************************

CODE    SEGMENT PARA PUBLIC 'CODE'
ASSUME  CS:CODE,DS:CODE,ES:NOTHING,SS:NOTHING

; All code and data of the assembly language calls is
; linked to the basic code segment.
;
; The following public declarations are recognized by
; the linker so that the routines can be accessed from
; outside of the object code.

        PUBLIC  FNTOUCH,OFFTOUCH,CURS,CLS,KEYLABEL,
                GCLEAR,GRAPH,GRAPHOFF,ALPHA
        PUBLIC  ALPHAOFF,GCURS,GCURSOFF,PENUP,PLOTU,
                PLOTD,MGCURS,DIR,SIZE
        PUBLIC LINETYPE

; *****************************************************
; This function declares a touchfield that gives the
; response resp$ to the; BASIC application when it is
; touched. A call of this function has to be done
; as follows :
;
; call fntouch (row%,col%,rowinc%,colinc%,resp$)
;
```

```
; where row% and col% are integers in the range of 0 to
79 resp. 0 to 47 that define the upper left corner of
; the touchfield. rowinc% and colinc% are the width and
; height of the touchfield. resp$ is the string that is
; handed over the application when the field is
; touched. These parameters are passed to the assembly
; language routine by means of the stack
; defined below :
;
; stack :

START_ROW       EQU     14
START_COLUMN    EQU     12
NUM_ROWS        EQU     10
NUM_COLUMNS     EQU     8
RESPONSE        EQU     6

STACK_SIZE      EQU      5*2

; code :

FNTOUCH PROC    FAR

        PUSH    BP
        MOV     BP,SP
        PUSH    DS
        MOV     SI,START_ROW[BP]
        MOV     AH,[SI]
        MOV     SI,START_COLUMN[BP]
        MOV     AL,[SI]
        MOV     CS:WORD PTR UPPER_CORNER,AX
        MOV     SI,NUM_ROWS[BP]
        ADD     AH,[SI]
        MOV     SI,NUM_COLUMNS[BP]
```

```
          ADD       AL,[SI]
          MOV       CS:WORD PTR LOWER_CORNER,AX
          MOV       SI,RESPONSE[BP]
          CLD
          LODSW
          MOV       CS:WORD PTR RES_LEN,AX
          LODSW
          MOV       CS:WORD PTR [RES_STRING],AX
          MOV       AX,DS
          MOV       CS:WORD PTR [RES_STRING+2],AX
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,OFFSET LEN_AIOS_CALL
          MOV       DX,OFFSET AIOS_CALL
          PUSH      CS
          POP       DS
          INT       21H
          POP       DS
          POP       BP
          RET       STACK_SIZE

; command buffer

AIOS_CALL:
               DW        32
RES_STRING               LABEL     WORD
               DW        2         DUP(?)
RES_LEN                  LABEL     WORD
               DW        ?
               DB        2
               DB        1
               DB        'J'
               DB        'B'
               DB        0
```

```
            DB          0
UPPER_CORNER            LABEL       WORD
            DW          ?
LOWER_CORNER            LABEL       WORD
            DW          ?
LEN_AIOS_CALL           EQU         $-AIOS_CALL

FNTOUCH ENDP

; ****************************************************
; This function erases all previously declared
; touchfields from the screen. In addition to that it
; is used to define the enhancement of the fields and
; the touch sensitivity. No additional parameter have
; to be passed to this routine. Therefore a call looks
; like this :
;
; call offtouch
;

OFFTOUCH  PROC      FAR
          PUSH      DS
          PUSH      CS
          POP       DS
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,OFFSET LEN_TURN_IT_OFF
          MOV       DX,OFFSET TURN_IT_OFF
          INT       21H
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,OFFSET LEN_REPORT_MODE
          MOV       DX,OFFSET REPORT_MODE
          INT       21H
          POP       DS
          RET

; command buffer

TURN_IT_OFF:
          DW        34
          DW        0FFFFH
LEN_TURN_IT_OFF     EQU       $-TURN_IT_OFF
REPORT_MODE:
          DW        36
          DW        12
          DW        2
LEN_REPORT_MODE     EQU       $-REPORT_MODE
OFFTOUCH  ENDP

; ***************************************************
; This function moves the alphanumeric curser to the
; desired location. Two integers have to be passed to
; this routine by means of a stack : the row (0-79) and
; the column (0-47) :
;
; call curs (col%,row%)
;
; stack :

COL       EQU       8
ROW       EQU       6
CURST     EQU       2*2

; code :

CURS      PROC      FAR
          PUSH      BP
          MOV       BP,SP
          PUSH      DS
          MOV       SI,COL[BP]
          MOV       AX,[SI]
          MOV       CS:WORD PTR KOLONNE,AX
          MOV       SI,ROW[BP]
          MOV       AX,[SI]
          MOV       CS:WORD PTR REIHE,AX

          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,7
          MOV       DX,OFFSET LOC_CALL
          PUSH      CS
          POP       DS
          INT       21H
          POP       DS
          POP       BP
          RET       CURST

; command buffer

LOC_CALL:
          DW        17
          DB        88H
KOLONNE   LABEL     WORD
          DW        ?
REIHE     LABEL     WORD
          DW        ?
CURS    ENDP

; *****************************************************
; The following routine erases the alphanumeric screen
; and places the cursor to the upper left of the
; screen. No paarmeters have to be passed to this
; routine.
;
; call cls
;
; code :

CLS         PROC        FAR
            PUSH        DS
            PUSH        CS
            POP         DS
            MOV         AX,CS
            MOV         CMDSEG,AX
            MOV         AX,4403H
            MOV         BX,1
            MOV         CX,8
            MOV         DX,OFFSET BATCH
            INT         21H
            POP         DS
            RET

; command buffer

CMDBUF   DB       16,0,'H',16,0,'J'
BATCH    DB       0,0
BUFLEN     DW   6
CMDOFF   DW       CMDBUF
CMDSEG   DW       ?
CLS      ENDP

; *******************************************************
; This routine is used to display application softkeys
; on the softkey label fields of the HP-150. No
; parameters have to be passed
```

```
;
; call keylabel
;

KEYLABEL            PROC        FAR
            PUSH    DS
            PUSH    CS
            POP     DS
            MOV     AX,4403H
            MOV     BX,1
            MOV     CX,2
            MOV     DX,OFFSET LAB_CALL
            INT     21H
            POP     DS
            RET
LAB_CALL:
            DW      11
KEYLABEL            ENDP


; *****************************************************
; The following routine erases the graphic screen.
Again, no parameters have to be passed to this routine.
; A call from MICROSOFT BASIC looks as follows :
;
; call gclear
;
; code :

GCLEAR              PROC        FAR
            PUSH    DS
            PUSH    CS
            POP     DS
            MOV     AX,4403H
```

```
          MOV       BX,1
          MOV       CX,2
          MOV       DX,OFFSET GCL_CALL
          INT       21H
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,2
          MOV       DX,OFFSET GTM_CALL
          INT       21H
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,2
          MOV       DX,OFFSET GDF_CALL
          INT       21H
          POP       DS
          RET

; command buffer

GCL_CALL:
          DB        1,4
GTM_CALL:
          DB        15,4
GDF_CALL:
          DB        38,4
GCLEAR          ENDP


; *****************************************************
; The following routine displays the graphic screen on
; the terminal.
;
; call grpah
;
```

```
GRAPH               PROC      FAR
          PUSH      DS
          PUSH      CS
          POP        DS
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,2
          MOV       DX,OFFSET GRA_CALL
          INT       21H
          POP       DS
          RET
GRA_CALL:
          DB        3,4
GRAPH               ENDP


; *******************************************************
; The following routine removes the graphic screen from
; the terminal.
;
; call graphoff
;
; code :

GRAPHOFF            PROC      FAR
          PUSH      DS
          PUSH      CS
          POP       DS
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,2
          MOV       DX,OFFSET GRO_CALL
          INT       21H
```

```
                POP         DS
                RET
GRO_CALL:
                DB          4,4
GRAPHOFF              ENDP


; ****************************************************
; The following routine displays the alphanumeric
; screen on the terminal.
;
; call alpha
;
; code :

ALPHA                  PROC        FAR
                PUSH        DS
                PUSH        CS
                POP         DS
                MOV         AX,4403H
                MOV         BX,1
                MOV         CX,2
                MOV         DX,OFFSET ALP_CALL
                INT         21H
                POP         DS
                RET
ALP_CALL:
                DB          5,4
ALPHA                 ENDP


; ****************************************************
; The following routine removes the alphanumeric screen
; from the terminal.
```

```
;
; call alphaoff
;
; code :

ALPHAOFF                PROC        FAR
            PUSH        DS
            PUSH        CS
            POP         DS
            MOV         AX,4403H
            MOV         BX,1
            MOV         CX,2
            MOV         DX,OFFSET ALO_CALL
            INT         21H
            POP         DS
            RET
ALO_CALL:
            DB          6,4
ALPHAOFF            ENDP


; ****************************************************
; The following routine is used to switch on the
; graphic cursor. No additional; parameters have to be
; passed.
;
; call gcurs
;
; code :

GCURS                   PROC        FAR
            PUSH        DS
            PUSH        CS
            POP         DS
```

```
                MOV        AX,4403H
                MOV        BX,1
                MOV        CX,2
                MOV        DX,OFFSET GCU_CALL
                INT        21H
                POP        DS
                RET
GCU_CALL:
        DB    7,4
GCURS                ENDP


; ****************************************************
; The following routine is used to switch off the
; graphic cursor. No additional parameters have to be
; passed.
;
; call gcursoff
;
; code :

GCURSOFF                   PROC       FAR
                PUSH       DS
                PUSH       CS
                POP        DS
                MOV        AX,4403H
                MOV        BX,1
                MOV        CX,2
                MOV        DX,OFFSET GCO_CALL
                INT        21H
                POP        DS
                RET
GCO_CALL:
                DB         08,4
```

```
GCURSOFF          ENDP


; *****************************************************
; The following routine is used to lift the graphic
; stylus. No additional parameters have to be pased.
;
; call penup
;

PENUP                   PROC        FAR
            PUSH        DS
            PUSH        CS
            POP         DS
            MOV         AX,4403H
            MOV         BX,1
            MOV         CX,2
            MOV         DX,OFFSET PEU_CALL
            INT         21H
            POP         DS
            RET
PEU_CALL:
            DW          39,4
PENUP             ENDP


; *****************************************************
; The following routine is used to move the graphic
; stylus to the desired location. The location of the
; graphic stylus is defined in pixels from the lower
; left of the screen. In the case of the HP 150, the
; maximal values for the x-position are from o to 640
; and for the y-position the  maximal values are from 0
; to 390. This routine does not draw a line from the
; former location to the new location entered by the
; integers x% and y%. A call from BASIC looks as
; follows :
;
; call plotu (x%,y%)
;
; Two integers are passed to the assembly language
     routine by a stack :

XKOR        EQU         8
YKOR        EQU         6
PLUST       EQU         2*2

; code :

PLOTU       PROC        FAR
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            MOV         SI,XKOR[BP]
            MOV         AX,[SI]
            MOV         CS:WORD PTR XWERT,AX
            MOV         SI,YKOR[BP]
            MOV         AX,[SI]
            MOV         CS:WORD PTR YWERT,AX

            MOV         AX,4403H
            MOV         BX,1
            MOV         CX,6
            MOV         DX,OFFSET PLU_CALL
            PUSH        CS
            POP         DS
            INT         21H
```

```
           POP        DS
           POP        BP
           RET        PLUST

; command buffer

PLU_CALL:
           DB         40,4
XWERT      LABEL      WORD
           DW         ?
YWERT      LABEL      WORD
           DW         ?
PLOTU    ENDP



; ******************************************************
; The following routine is used to draw a line of the
; active linetype (see below) from the current location
; to the new location of the graphic stylus defined by
; the integers x% and y%. The location of the graphic
; atylus is defined in pixels from the lower left of
; the screen. In the case  of the HP 150, the maximal
; values for the x-position are from 0 to 640 and  for
; the y-position the maximal values are from 0 to 390.
; A call from BASIC  looks as follows :
;
; call plotd (x%,y%)
;
; Two integers are passed to the assembly language
     routine by a stack :

XKORD      EQU        8
YKORD      EQU        6
```

```
PLDST     EQU       2*2

; code :

PLOTD     PROC      FAR
          PUSH      BP
          MOV       BP,SP
          PUSH      DS
          MOV       SI,XKORD[BP]
          MOV       AX,[SI]
          MOV       CS:WORD PTR XWERTD,AX
          MOV       SI,YKORD[BP]
          MOV       AX,[SI]
          MOV       CS:WORD PTR YWERTD,AX

          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,6
          MOV       DX,OFFSET PLD_CALL
          PUSH      CS
          POP       DS
          INT       21H
          POP       DS
          POP       BP
          RET       PLDST

; command buffer

PLD_CALL:
          DB        44,4
XWERTD    LABEL     WORD
          DW        ?
YWERTD  LABEL  WORD
          DW        ?
```

```
PLOTD   ENDP


; ****************************************************
; The following routine is used to move the graphic
; cursor the the desired location. The location of the
; graphic cursor is defined in pixels from the lower
; left of the screen. In the case of the HP 150, the
; maximal  values for the x-position are from o to 640
; and for the y-position the  maximal values are from 0
; to 390. A call from BASIC looks as follows :
;
; call mgcurs (col%,row%)
;
; Two integers are passed to the assembly language
     routine by a stack :

XKORC      EQU        8
YKORC      EQU        6
MGCST      EQU        2*2

; code :

MGCURS     PROC       FAR
           PUSH       BP
           MOV        BP,SP
           PUSH       DS
           MOV        SI,XKORC[BP]
           MOV        AX,[SI]
           MOV        CS:WORD PTR XWERTC,AX
           MOV        SI,YKORC[BP]
           MOV        AX,[SI]
           MOV        CS:WORD PTR YWERTC,AX
```

```
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,6
          MOV       DX,OFFSET MGC_CALL
          PUSH      CS
          POP       DS
          INT       21H
          POP       DS
          POP       BP
          RET       MGCST

; command buffer

MGC_CALL:
          DB        11,4
XWERTC    LABEL     WORD
          DW        ?
YWERTC  LABEL  WORD
     D    W         ?
MGCURS  ENDP


; *******************************************************
; The following routine is used to define the direction
; of the graphic text. There are only four possible
; directions : 1=horizontal, 2=vertical, 3=horizontal
; from right to left and 4=vertical from top to bottom.
; The single variable is passed by the stack :
;
; call dir (dir%)
;
DIRW      EQU       6
DIRST     EQU       1*2
```

```
; code :

DIR         PROC        FAR
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            MOV         SI,DIRW[BP]
            MOV         AX,[SI]
            MOV         CS:WORD PTR WDIR,AX

            MOV         AX,4403H
            MOV         BX,1
            MOV         CX,4
            MOV         DX,OFFSET DIR_CALL
            PUSH        CS
            POP         DS
            INT         21H
            POP         DS
            POP         BP
            RET         DIRST
DIR_CALL:
            DB          30,4
WDIR        LABEL       WORD
            DW          ?
DIR       ENDP

; ******************************************************
; The following routine is used to define the size of
; the graphic text in units of pixels * 128. Two
; variables have to be passed by a stack : The x-size,
; and the y-size. A call from BASIC looks like this :
;
; call size (x%,y%)
;
```

```
XSIZE     EQU       8
YSIZE     EQU       6
SIZST     EQU       2*2

; code :

SIZE       PROC     FAR
          PUSH      BP
          MOV       BP,SP
          PUSH      DS
          MOV       SI,XSIZE[BP]
          MOV       AX,[SI]
          MOV       CS:WORD PTR XDIM,AX
          MOV       SI,YSIZE[BP]
          MOV       AX,[SI]
          MOV       CS:WORD PTR YDIM,AX
          MOV       AX,4403H
          MOV       BX,1
          MOV       CX,6
          MOV       DX,OFFSET SIZ_CALL
          PUSH      CS
          POP       DS
          INT       21H
          POP       DS
          POP       BP
          RET       SIZST
SIZ_CALL:
          DB        29,4
XDIM      LABEL     WORD
          DW        ?
YDIM    LABEL   WORD
          DW        ?
SIZE    ENDP
```

```
; ****************************************************
; The following routine is used to select a linetype
; out of several linetypes defined in the BIOS of the
; HP-150. One integer in the range from 0 to 10 has to
; be passed to this routine.
;
; call linetype(lt%)

LINIE       EQU         6
LINST       EQU         1*2

LINETYPE                PROC        FAR
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            MOV         SI,LINIE[BP]
            MOV         AX,[SI]
            MOV         CS:WORD PTR WLIN,AX

            MOV         AX,4403H
            MOV         BX,1
            MOV         CX,4
            MOV         DX,OFFSET LIN_CALL
            PUSH        CS
            POP         DS
            INT         21H
            POP         DS
            POP         BP
            RET         LINST
LIN_CALL:
            DB          18,4
WLIN        LABEL       WORD
```

```
          DW          ?
LINETYPE           ENDP

; ******************************************************

CODE    ENDS

        END

; ******************************************************

E*******************************************************
:    dos Funktionen            B.Michel
E*******************************************************
;
read_kbd_echo   macro           ;holt Zeichen von
     mov        ah,1            ;keyboard und zeigt
     int        33              ;es auf dem Bildschirm
     endm                       ;an ^C ergibt int 23H
;
display_char    macro character;zeigt Zeichen auf dem
     mov        dl,character    ;Bildschirm
     mov        ah,2
     int        33
     endm
;
aux_input       macro           ;wartet auf Zeichen vom
     mov        ah,03H          aux input dev und gibt
     int        21H             das Zeichen in al
     endm
;
aux_output      macro character;sendet das Zeichen in
     mov        dl,character    ;dl an das aux output
     mov        ah,04H          ;device
```

```
          int       21H
          endm
;
print_char          macro character;printet das Zeichen
          mov       dl,character   ;in DL auf dem
          mov       ah,05          ;standard printer
          int       21H
          endm
;
dir_con_io          macro switch   ;wenn ein Zeichen
          mov       dl,switch      ;eingetippt wurde ist
          mov       ah,06H         ;das Zero flag 0 und sa
          int       21H            ;Zeichen in AL sonst ist
          endm                     ;das Zero Flag 1
;
read_kbd            macro          ;wartet auf Zeichen und
          mov       ah,08H         ;gibt es in AL
          int       21H
          endm
;
display             macro string   ;Offset adresse in DX von
          mov      dx,offset string;DS ende mit $
          mov       ah,09H
          int       21H
          ENDM
;
select_disk         macro disk     ;drive A,B,C ....
          mov       dl,disk[-65]
          mov       ah,OEH
          int       21H
          endm
;
open                macro fcb      ;AL wird O wenn das file
          mov       dx,offset,fcb  ;gefunden wurde
          mov       ah,OFH
          int       21H
          endm
;
close               macro fcb
          mov       dx,offset,fcb
          mov       ah,10H
          int       21H
          endm
;
search_first        macro,fcb          ;fcb ist ein
          mov       dx,offset,fcb      ;ungeoeffneter
          mov       ah,11H             ;Filecontrol block:
          int       21H                fcb  db  2,"?????"
          endm                         ;    db  25 dup (?)
;
search_next         macro fcb          ;gefunden -AL=0 nicht
          mov       dx,offset,fcb      ;gef.-AL=FFH ein fcb
          mov       ah,12H             ;wird bei der disk
          int       21H                ;transfer adresse
          endm                         ;geschrieben
;
set_dta             macro buffer       ;
          mov       dx,offset,buffer
          mov       ah,1AH
          int       21H
          endm
;
get-date            macro              ;CX=Jahr
          mov       ah,2AH             ;DH=Monat 1-12
          int       21H                ;DL=Tag 1-31
          endm                         ;AL=Wochentag 0=Sonntag
;
get_time            macro              ;CH=Stunde 0-23
```

```
          mov        ah,2CH            ;CL=Minuten 0-59
          int        21H               ;DH=Sekunden
          endm
;
get disk transfer address
          mov        ah,2AH
          int        21H
          es:bx hat nun die laufende DMAtransfer addresse
;
get_dfree            macro             ;0=Default 1=A
          mov        ah,36H            ;bx : Anzahl Einheiten
          mov        dl,drive          ;cx : Bytes per Sector
          int        21H               ;ax :Sectoren pro Einheit
          endm
;
parse                macro  string,fcb
          mov        si,offset string
          mov        di,offset fcb
          push       es
          push       ds
          pop        es
          mov        al,0FH
          mov        ah,29H
          int        21H
          pop        es
          endm
;
```

APPENDIX G

Start Overlay Autost

```
50 ! KINETIK B.MICHEL
90 ! STORE "Autost.KSYS"
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
    L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
130 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
    D3$[20],D4$[20],B9$[8]
200 SHORT Y0,Y1,Y2,Y3,X0,X1,X2,X3,L1,L2,X4,X5,X6,
    X7,Y4,Y5,Y6,Y7
210 DIM E3$[30],E4$[30],E5$[60]
220 DIM H1$[60]
250 ON ERROR GOTO 260
255 IF C(8)=1 THEN 4000
260 CLEAR @ C(8)=1 @ D2=10 @ C(9)=0 @ C(6)=1 @ C(15)=0
    @ GOSUB 265 @ GOTO 295
265 DISP "*********************************"
270 DISP "***  Messprogram for the     ***" @ DISP "***
    autom. determination of ***"
275 DISP "***  Michaelis constants     ***"
280 DISP "***  B.Michel       Version   **" @ DISP "***
    05. Nov. 86    ***"
290 DISP "*********************************" @ RETURN
295 FOR I=1 TO 41 @ T1(I)=0 @ T2(I)=0 @ NEXT I
297 FOR I=1 TO 20 @ T(I)=0 @ L(I)=0 @ B(I)=0 @ W(I)=0 @
    E(I)=0 @ NEXT I
300 ! ***W ARRAY**
305 W(1)=50 @ W(2)=5 @ W(3)=2.5 @ W(4)=100 @ W(5)=1
307 ! MAX. Volumen
310 W(6)=50 @ W(7)=5 @ W(8)=2.5 @ W(9)=100 @ W(10)=1
312 ! Max. Fuellung
315 W(11)=1 @ W(12)=.5 @ W(13)=.25 @ W(14)=1 @ W(15)=1
```

```
317 ! Menge pro Tastendruck
320 W(16)=1 @ W(17)=.1 @ W(18)=.05 @ W(19)=1 @
     W(20)=.01
322 ! Geschwindigkeit
325 W1(1)=.4 @ W1(2)=.05 @ W1(3)=.05 @ W1(4)=60000 @
     W1(5)=60
327 ! Stopfl. Mengen und Anz. Uml. und Beschl.
330 C(1)=0 @ C(2)=0 @ C(3)=0 @ C(4)=0 @ C(5)=0
332 ! Counter
340 ! **T-ARRAY***
342 T(1)=548 @ T(2)=552 @ T(3)=.1 @ T(4)=150 @ T(5)=20
     @ T(6)=4 @ T(7)=10
344 ! Wellenl. 1 und 2 Ber.1 , Enzym und Substrat-Konz,
     Anz. Punkte Anfangs- und Endzeit B1
346 T(8)=3 @ T(9)=1 @ T(10)=580 @ T(11)=586 @ T(12)=416
     @ T(13)=420 @ T(14)=30
348 ! Anz. Rep., Datenforamt, Wellenlaengen 1 und 2
     Norm., Wellenl. 1 und 2 Ber. 2,Max. Substr.
350 T(15)=27.6 @ T(16)=60 @ T(17)=-16.3 @ T(18)=-43
352 ! Epsilon Substrat Bereich 1 , Epsilon Substr.
     Bereich 2 und Delta Epsilon 1 und 2
354 T(19)=1 @ T(20)=1 @ T(0)=T(8)
356 ! Art der Reihe    /  Zus.  Repetitionen bei Punkt
     1
358 T1$="..............." @ T1$=T1$&T1$&T1$&T1$
359 ! ***B-ARRAY*******
360 B(1)=1 @ B(2)=1 @ B(3)=.5 @ B(4)=0 @ B(5)=.5
362 ! Max zul. Stdev./ Stdev Einzelmsg / [E] im Assay /
     [E]auto / [E]im Assay Vorwahl
364 B(6)=10 @ B(7)=.5
366 ! [S] tatsachlich / Assay Volumen
368 B(11)=.5 @ B(12)=.5 @ B(13)=1 @ B(14)=4
370 ! 11 : Intervall und 12 : Integrationszeit, 13/14
     Messbereich 2
```

```
376 ! L(1) BIS L(10)= MESSWELLENLAENGEN
377 FOR I=1 TO 20 @ L(I)=1 @ NEXT I
378 E1$="..................." @ E2$="Cytochrome c"
380 D4$="KAON0000       0" @ D1$="A" @ D2=10
386 ! ******Plot Defaults******
388 A(1,1)=0 @ A(1,2)=100 @ A(1,3)=20 @ A(1,4)=0 ! TN
     vs. Zeit
390 A(2,1)=-.01 @ A(2,2)=.001 @ A(2,3)=.001 @ A(2,4)=0
     ! Delta OD Plot
392 A(3,1)=-.1 @ A(3,2)=1.5 @ A(3,3)=.2 @ A(3,4)=.1 !
     OD Plot
398 ! ****Ende Plot defaults **
400 DISP @ FLIP @ DISP "Please enter the date and the
     name of the diskette"
401 DISP "eg.  30.Okt. 86,Michel"
402 DISP "The two inputs have to be se-    parated by a
     comma";@ INPUT D3$,B9$@ FLIP
403 ON ERROR GOTO 410
404 GOSUB 2000 @ GOTO 3900
410 DISP "File ";D$;" not found" @ DISP "Please select
     :" @ ON ERROR GOTO 4000
412 DISP "1 = new read" @ DISP "2 = rename" @ DISP "3 =
     new Directory"
414 DISP "4 = initialising" @ DISP "5 = show Directory
     " @ DISP "9 = Typing error"
420 INPUT H2
422 IF H2<2 OR H2>5 THEN 400
424 IF H2=3 THEN GOSUB 2200 @ GOTO 3900
426 DISP "Please enter Drive (0/1/2/3/4) 0=:D700
     1=:D701  4=:D710" @ INPUT H1
430 IF H2=4 THEN 1650
435 ON ERROR GOTO 450 @ H1$=":D700:D701:D702:D703:D710"
     @ A$=H1$[H1*5+1,H1*5+5]
```

```
440 IF H2=5 THEN CLEAR @ DISP "Press (+cont)" @ CAT A$
     @ PAUSE
442 IF H2=5 THEN 412
443 ON ERROR GOTO 400
444 IF H2=2 THEN VOLUME A$ IS B9$ @ GOTO 402
448 BEEP @ GOTO 400
450 IF ERRN=130 THEN DISP "Please check the Drive " @
     DISP A$ @ BEEP
455 DISP "****** Error *******" @ DISP "******
     ";ERRN;" ******" @ BEEP
460 GOTO 400
500 ON ERROR GOTO 9000 @ CLEAR
505 DISP "******* Main programm **********" @ DISP
     "***** Mess Parameter Menu 1 ****"
510 DISP E1$;TAB(20);T(3);"nM (E)" @ DISP
     E2$;TAB(20);T(4);"M (S)"
515 DISP "[S]max in Asssay";TAB(22);T(14);"M"
520 DISP "Range  1 from :";T(1);" to";T(2);"nm" @ DISP
     "Range  2 from :";T(12);" to";T(13);"nm"
525 DISP "Int. ref. form:";T(10);" to";T(11);"nm" @
     DISP "Epsilon   Range 1  and Range 2"
530 DISP "Substrate:";TAB(12);T(15);TAB(18);T(16);"1/mM
     cm"
535 DISP "Product  :";TAB(12);T(15)+T(17);TAB(18);T(16)
     +T(18);"1/mM cm"
540 DISP "No. of Points";TAB(24);T(5) @ DISP "No. of
     Repetitons";TAB(24);T(8);T(20)
550 ON KEY# 1,"Exit" GOTO 4000
552 ON KEY# 2,"Points" GOTO 840
554 ON KEY# 3,"Range" GOTO 760
556 ON KEY# 4,"Page2" GOTO 600
558 ON KEY# 5,"Conc." GOTO 880
560 ON KEY# 6,"NameE" GOTO 690
562 ON KEY# 7,"[S]max" GOTO 890
```

```
564 ON KEY# 8,"Epsilon" GOTO 700
566 KEY LABEL @ GOTO 550
600 CLEAR @ DISP "***** Mess Parameter Menu 2 ****"
605 DISP "Durations:"
610 DISP "at [S]max :  ";T(6);" to";T(7);"sec" @ DISP
     "at [S] min:  ";B(13);" to";B(14);"sec"
615 DISP "Max Standarddev.Av";B(1);" EzMsg";B(2);"%" @
     DISP "Data form: ";T(9);"  ";
620 IF T(19)=1 THEN DISP "Geom. Row"
622 IF T(19)=2 THEN DISP "Arithm. Row"
624 IF T(19)=3 THEN DISP "Mixed Row"
626 IF T(19)=4 THEN DISP "Reciprocal Row"
630 DISP "Y-Axis TN vs. [S] Plot :" @ DISP A(1,1);"
     to";A(1,2);"d";A(1,3);"d0";A(1,4)
635 DISP "Y-Axis delta OD vs Time Plot:" @ DISP
     A(2,1);" to";A(2,2);"d";A(2,3);"d0";A(2,4)
640 DISP "Y-Axis OD vs Time Plot:" @ DISP A(3,1);"
     to";A(3,2);"d";A(3,3);"d0";A(3,4)
670 ON KEY# 1,"Row" GOTO 995
672 ON KEY# 2,"Stdev" GOTO 920
674 ON KEY# 3,"Durat" GOTO 740
676 ON KEY# 4,"Page1" GOTO 500
678 ON KEY# 5,"TN Sca" GOTO 960
680 ON KEY# 6,"OD Sca" GOTO 940
682 ON KEY# 7,OD Sca" GOTO 980
684 ON KEY# 8,"Dform" GOTO 720
686 KEY LABEL @ GOTO 670
690 CLEAR @ DISP "Enter name of enzyme" @ FLIP @ INPUT
     E1$
695 DISP "Enter name of substrate" @ INPUT E2$@ FLIP @
     GOTO 500
700 CLEAR @ DISP "Enter the extiction coefficient of
     substrate and product at"
```

```
702 DISP T(1);" to  ";T(2);" nm  in (1/mM cm)" @ INPUT
     T(15),H1@ T(17)=H1-T(15)
704 DISP "Enter the extiction coeffizient of substrate
     and product at"
706 DISP T(12);" to  ";T(13);" nm  in (1/mM cm)" @
     INPUT T(16),H1@ T(18)=H1-T(16) @ GOTO 500
720 IF T(9)=1 THEN T(9)=2 @ GOTO 600
725 IF T(9)=2 THEN T(9)=3 @ GOTO 600 ELSE T(9)=1 @ GOTO
600
740 CLEAR @ DISP "Enter duration at [S]max.";@ INPUT
     T(6),T(7)
741 DISP "Enter duration at [S]min.";@ INPUT
     B(13),B(14)
742 DISP "Enter measur interval and inte- gration
     time";@ INPUT L2,L1
744 IF L1>L2 OR L1 MOD .1#0 OR L2 MOD .1#0 THEN GOTO
742
745 IF B(13)>B(14) THEN GOTO 741
746 IF T(7)/L2>80 OR T(7) MOD L2#0 THEN GOTO 740
747 IF T(7)/L2>40 AND T(9)>1 THEN GOTO 740
750 IF B(14)>T(7) THEN GOTO 740
755 B(15)=B(13) @ B(16)=T(7) @ B(11)=L1 @ B(12)=L2 @
     GOSUB 5100 @ GOSUB 5200 @ GOTO 600
760 CLEAR @ DISP "Enter wavelength Range 1";@ INPUT
     T(1),T(2)@ GOSUB 4800
800 DISP "Enter wavelength range 2";@ INPUT
     T(12),T(13)@ GOSUB 4800
810 DISP "Enter Wavelength Range for int. reference";
820 INPUT T(10),T(11)@ GOSUB 4800 @ GOTO 500
840 CLEAR @ DISP "Enter Number of Points";@ INPUT T(5)
842 DISP "Enter number of repetitions and additional
     repetions at [S]min.";@ INPUT T(8),T(20)
845 IF T(5)*T(8)>100 OR T(8)>10 OR T(5)>40 OR T(5)<0 OR
     T(8)<2 THEN GOTO 840
```

```
850 GOTO 500
880 CLEAR @ DISP "Enter conc. of enzyme (in nM)   and
     substrate (in M)";
885 INPUT T(3),T(4)@ GOTO 500
890 CLEAR @ DISP "Maximal substrate conzentration in
     Assay (in M)";@ INPUT T(14)
900 IF T(14)<T(4)/15 OR T(14)>T(4)/4 THEN GOTO 890
910 GOSUB 5000 @ GOTO 500
920 CLEAR @ DISP "Enter the max.standarddeviation of
     the average in %";@ INPUT B(1)
925 DISP "Enter the max. nonlinearity of  single
     measurements in %";@ INPUT B(2)
930 IF B(1)<1 OR B(1)>100 OR B(2)<1 OR B(2)>100 THEN
     GOTO 920 ELSE GOTO 600
940 CLEAR @ DISP "Enter Y-min, Y-max, delta Y and delta
     Yo for OD Plot"
945 INPUT A(3,1),A(3,2),A(3,3),A(3,4)
947 IF A(3,2)<A(3,1) OR A(3,3)>A(3,2)-A(3,1) OR
     A(3,4)<0 THEN GOTO 940
950 GOSUB 5100 @ GOTO 600
960 CLEAR @ DISP "Enter Y-min, Y-max, delta Y and delta
     Yo for V Plot";
965 INPUT A(1,1),A(1,2),A(1,3),A(1,4)
967 IF A(1,2)<A(1,1) OR A(1,3)>A(1,2)-A(1,1) OR
     A(1,4)<0 THEN GOTO 960
970 GOSUB 5000 @ GOTO 600
980 CLEAR @ DISP "Enter Y-min, Y-max, delta Y and delta
     Yo forOD Plot";
985 INPUT A(2,1),A(2,2),A(2,3),A(2,4)
987 IF A(2,2)<A(2,1) OR A(2,3)>A(2,2)-A(2,1) OR
     A(2,4)<0 THEN GOTO 980
990 GOSUB 5200 @ GOTO 600
995 IF T(19)=1 THEN T(19)=2 @ GOTO 600
996 IF T(19)=2 THEN T(19)=3 @ GOTO 600
```

```
997 IF T(19)=3 THEN T(19)=4 @ GOTO 600 ELSE T(19)=1 @
     GOTO 600
1000 ! ******** Syringe P******
1010 CLEAR @ DISP "**** Syringe Parameter Menu ****" @
     DISP
1030 DISP "     Volume  Fill   Mpk  ml/sec"
1040 DISP "Syr1:";TAB(8);W(1);TAB(14);W(6);TAB(20);
     W(11);TAB(26);W(16)
1045 DISP "Syr2:";TAB(8);W(2);TAB(14);W(7);TAB(20);
     W(12);TAB(26);W(17)
1050 DISP "Syr3:";TAB(8);W(3);TAB(14);W(8);TAB(20);
     W(13);TAB(26);W(18)
1055 DISP "Tray:";TAB(8);W(4);TAB(14);W(9);TAB(20);
     W(14);TAB(26);W(19)
1060 DISP "Need:";TAB(8);W(5);TAB(14);W(10);TAB(20);
     W(15);TAB(26);W(20)
1070 DISP "Stopfl.";W1(1);W1(2);W1(3)
1075 DISP "Accel.";W1(5);" Schrr ";W1(4)
1080 DISP "Volume per Assay";B(7);"ml"
1150 ON KEY# 1,"Exit" GOTO 4000
1160 ON KEY# 2,"Stpfl" GOTO 1350
1170 ON KEY# 3,"Accel" GOTO 1300
1180 ON KEY# 4,"Needle" GOTO 1480
1190 ON KEY# 5,"Syr.1" GOTO 1400
1200 ON KEY# 6,"Syr.2" GOTO 1420
1210 ON KEY# 7,"Syr.3" GOTO 1440
1220 ON KEY# 8,"Tray" GOTO 1460
1230 KEY LABEL @ GOTO 1150
1300 CLEAR @ DISP "Enter Acceleration for stoppped
     flow:";W1(5);" neu";@ INPUT W1(5)
1310 CLEAR @ DISP "Enter No. of cycles for
     stopppedFlow:";W1(4);" neu";@ INPUT W1(4)
1320 GOTO 1000
```

```
1350 CLEAR @ DISP "Enter the amounts for stopped  Flow
     former :";W1(1);W1(2);W1(3);"ml"
1355 DISP " new ";@ INPUT W1(1),W1(2),W1(3)@
     B(7)=W1(1)+W1(2)+W1(3)
1360 GOTO 1000
1400 CLEAR @ DISP "Enter parameter for syringe
     1:",W(1);W(6);W(11);W(16);" new";
1410 INPUT W(1),W(6),W(11),W(16)@ GOTO 1000
1420 CLEAR @ DISP "Enter parameter for syringe
     2",W(2);W(7);W(12);W(17);" new";
1430 INPUT W(2),W(7),W(12),W(17)@ GOTO 1000
1440 CLEAR @ DISP "Enter parameter forsyringe
     3:",W(3);W(8);W(13);W(18);" new";
1450 INPUT W(3),W(8),W(13),W(18)@ GOTO 1000
1460 CLEAR @ DISP "Enter parameters for Tray
     :",W(4);W(9);W(14);W(19);" new";
1470 INPUT W(4),W(9),W(14),W(19)@ GOTO 1000
1480 CLEAR @ DISP "Enter parameters for needle
     :",W(5);W(10);W(15);W(20);" new";
1490 INPUT W(5),W(10),W(15),W(20)@ GOTO 1000
1600 CLEAR @ DISP "Enter drive: 0=':D700' 1=':D701' and
     4=:D710'";@ INPUT H1
1602 IF H1=0 THEN MASS STORAGE IS ":D700" @ GOTO 1605
1604 IF H1=1 THEN MASS STORAGE IS ":D701" ELSE MASS
     STORAGE IS ":D710"
1605 CAT @ DISP "Enter filename to erase or
     (PACK/INIT)"
1610 INPUT H1$
1620 IF H1$="PACK" THEN PACK @ GOTO 4000
1625 IF H1$="INIT" THEN GOTO 1650
1630 IF H1$="N" THEN GOTO 4000
1635 IF H1$="" THEN GOTO 4000
1640 PURGE H1$ @ GOTO 1610
```

```
1650 IF H1=0 THEN DISP "Systemdisk must not be
     initialized" @ WAIT 2000 @ GOTO 4000
1657 DISP "Are you sure to erase all data  on this
     diskette (J/N)";@ INPUT H1$
1658 IF H1$#"J" THEN 4000
1660 DISP "Enter new name of diskette" @ INPUT H2$
1665 DISP "Please wait about 2 minutes"
1670 H1$=":D700:D701:D702:D703:D710" @ A$=H1$[H1*5+1,
     H1*5+5]
1675 INITIALIZE H2$,A$,20
1690 GOSUB 2200 @ GOTO 4000
2000 CLEAR @ DISP "The active Data diskette is  :  " @
     DISP "*****   ";B9$;"   *****"
2010 DISP "It contains data about   :   "
2025 D$="Inhalt."&B9$
2030 ASSIGN# 1 TO D$
2040 ON ERROR GOTO 2080
2050 READ# 1 ; H1$@ DISP H1$
2060 GOTO 2050
2080 ASSIGN# 1 TO *
2090 RETURN
2200 D$="Inhalt."&B9$ @ ON ERROR GOTO 2205 @ PURGE D$
2205 ON ERROR GOTO 9000 @ CREATE D$,2,256
2210 ASSIGN# 1 TO D$
2220 ON ERROR GOTO 2290
2230 CLEAR @ DISP "Please enter max . 12 lines of
     comment " @ I=1
2250 DISP "Zeile:";I @ FLIP @ INPUT H1$@ FLIP
2255 IF LEN(H1$)=0 OR LEN(H1$)>32 OR I>12 THEN 2290
2260 PRINT# 1 ; H1$ @ I=I+1 @ GOTO 2250
2290 ON ERROR GOTO 9000
2295 ASSIGN# 1 TO *
2390 RETURN
2395 ON ERROR GOTO 4000
```

```
3900 D$="DEFAULT.KSYS"
3905 ON ERROR GOTO 4000
3910 GOTO 4410
4000 ! ***MAIN MENU***
4020 CLEAR @ GOSUB 265
4100 DISP "Please select a softkey:"
4110 DISP "all other keys stop the program"
4130 ON ERROR GOTO 9000
4150 ON KEY# 1,"Km Msg" GOTO 4250
4160 ON KEY# 2,"Dacom" GOTO 4320
4170 ON KEY# 3,"ReadV" GOTO 4400
4180 ON KEY# 4,"MeasPa" GOTO 500
4190 ON KEY# 5,"Direct" GOTO 1600
4200 ON KEY# 7,"Sto Va" GOTO 4500
4210 ON KEY# 6,"Test M" GOTO 4340
4220 ON KEY# 8,"SyrinPa" GOTO 1000
4230 KEY LABEL @ GOTO 4230
4250 CLEAR @ DISP "***** Checking parameters *****"
4270 GOSUB 4600 ! ASSIGN
4280 GOSUB 5000 @ GOSUB 5100 @ GOSUB 5200 ! GENERATE
     PLOTS
4290 ON ERROR GOTO 4295 @ CREATE "TEMP1.KSYS",10,768 @
     CREATE "TEMP.KSYS",40,768
4295 SECURE "TEMP1.KSYS","AA",3 @ SECURE
     "TEMP.KSYS","AA",3
4300 REM
4306 ON ERROR GOTO 4310 @ )
4307 FOR I=1 TO 99 @ I 1140 ON TIMER# NEXT
4310 ON ERROR GOTO 9000
4311 DISP "**********************************" @ DISP
"******* The Application  *******" @ D2=0
4312 IF T(9)=1 THEN E3$="KIN 2.KSYS" @ DISP "Enzyme
     screening" @ D2=1
```

```
4313 IF T(9)=2 THEN E3$="KIN 1.KSYS" @ DISP "First
     order const" @ D2=2
4314 IF T(9)=3 THEN E3$="KININ.KSYS" @ DISP "Inhibition
```

```
4313 IF T(9)=2 THEN E3$="KIN 1.KSYS" @ DISP "First
     order const" @ D2=2
```

```
4314 IF T(9)=3 THEN E3$="KININ.KSYS" @ DISP "Inhibition
     screening" @ D2=3
4315 IF T(9)=4 THEN E3$="KINSU.KSYS" @ DISP "Substrate
     screening" @ D2=4
4316 IF T(9)=5 THEN E3$="KINTWO.KSYS" @ DISP "Two
     substrate determination" @ D2=5
4317 DISP "is loaded" @ DISP @ DISP "Please wait
     approx. 20 seconds"
4318 IF C(15)=0 THEN DISP "**** Initializing ASSAYOMATE
     ***" @ CHAIN "ASINIT.KSYS"
4319 D2=0 @ CHAIN E3$
4320 CLEAR @ DISP "The Datatransferprogram is
     loaded                                      "
4330 CHAIN "DACOM.KSYS"
4340 CLEAR @ DISP "The Application Testmeasure"
4360 GOSUB 4600 ! ASSIGN
4370 GOSUB 5000 @ GOSUB 5100 @ GOSUB 5200 ! GENERATE
     PLOTS
4380 D2=0 @ E3$="KONZMSG.KSYS" @ GOTO 4317
4400 CLEAR @ CAT ".KSYS"
4405 DISP "Please enter the Filename" @ INPUT D$@
     D$=D$&".KSYS"
4410 ASSIGN# 1 TO D$
4412 ON ERROR GOTO 4450
4420 READ# 1 ; D1$,H1,D4$,D4$,E1$,E2$,T1$,T(),L(),B(),
     A(,),W(),W1()
4430 ASSIGN# 1 TO * @ ON ERROR GOTO 9000
4440 GOSUB 5000 @ GOSUB 5100 @ GOSUB 5200
4445 GOTO 4000
```

```
4450 DISP "Old File : Please update Syringe Parameters"
4490 GOTO 4430
4500 CLEAR @ CAT ".KSYS"
4505 DISP "Please enter the Filename" @ INPUT D$
4507 D$=D$&".KSYS"
4510 GOSUB 4600 ! ASSIGN
4520 ON ERROR GOTO 4565
4560 CREATE D$,5,256
4565 ON ERROR GOTO 9000 @ ASSIGN# 1 TO D$
4570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B(),
     A(,),W(),W1()
4580 ASSIGN# 1 TO *
4590 GOTO 4000
4600 ! ****ASSIGN ***
4602 GOSUB 4800
4605 FOR I=1 TO 10 @ L(I)=178 @ NEXT I @ I1=0
4610 FOR I=T(1) TO T(2) STEP 2
4620 I1=I1+1 @ L(I1)=I
4630 NEXT I
4640 FOR I=T(12) TO T(13) STEP 2
4650 FOR J=1 TO 10 @ IF I=L(J) THEN GOTO 4665
4655 IF I1=10 THEN BEEP @ DISP "ZU GROSSER
     WELLENLAENGENBEREICH" @ GOTO 500
4660 I1=I1+1 @ L(I1)=I
4665 NEXT I
4670 FOR I=T(10) TO 820 STEP 2
4672 FOR J=1 TO 10 @ IF I=L(J) THEN GOTO 4680
4675 I1=I1+1 @ L(I1)=I
4677 IF I1=10 THEN GOTO 4690
4680 NEXT I
4690 H2$=" " @ REM
4700 IF T(3)<.0001 OR T(3)>10000 OR T(4)<.001 OR
     T(4)>100000 THEN H1=3 @ H2=4 @ GOSUB 4795
```

```
4705 IF T(5)<2 OR T(5)>40 OR T(8)<2 OR T(8)>10 OR
     T(5)*T(8)>200 THEN H1=5 @ H2=8 @ GOSUB 4795
4710 IF T(9)>3 OR T(9)<1 OR T(19)<1 OR T(19)>5 THEN
     H1=9 @ H2=19 @ GOSUB 4795
4715 IF T(14)>T(4)/5 OR T(14)<T(4)/100 THEN H1=14 @
     H2=0 @ GOSUB 4795
4720 IF T(6)<.2 OR T(6)>100 OR T(7)<T(6)+1 OR T(7)>2000
     THEN H1=6 @ H2=7 @ GOSUB 4795
4725 IF T(15)<.001 OR T(15)>1000 OR T(16)<.001 OR
     T(16)>1000 THEN H1=15 @ H2=16 @ GOSUB 4795
4730 IF T(17)<-100 OR T(17)>100 OR T(18)<-100 OR
     T(18)>100 THEN H1=17 @ H2=18 @ GOSUB 4795
4735 IF B(1)<1 OR B(1)>100 OR B(2)<1 OR B(2)>100 THEN
     H1=21 @ H2=22 @ GOSUB 4795
4750 IF B(7)<.1 OR B(7)>10 THEN H1=27 @ H2=28 @ GOSUB
     4795
4760 IF B(11)<.2 OR B(11)>20 OR B(12)<B(11) OR
     B(12)>100 THEN H1=31 @ H2=32 @ GOSUB 4795
4765 IF B(13)<.2 OR B(13)>B(14) OR B(14)>T(7) OR
     B(14)>1000 THEN H1=33 @ H2=7 @ GOSUB 4795
4780 IF H2$="F" THEN BEEP @ WAIT 2000 @ GOTO 4000
4790 RETURN
4795 IF H1<20 THEN H1$="T" ELSE H1=H1-20 @ H2=H2-20
4796 IF H1<20 THEN H1$="B" ELSE H1=H1-20 @ H2=H2-20 @
     H1$="W"
4797 IF H2$=" " THEN CLEAR @ H2$="F"
4798 BEEP @ DISP "Variables ";H1$;H1;" or ";H2;"are
     invalid"
4799 WAIT 2000 @ RETURN
4800 REM *** TEST**
4810 IF T(1)>820 OR T(1)<180 OR T(1) MOD 2#0 THEN
     T(1)=200 @ GOTO 4900
4815 IF T(2)>820 OR T(2)<180 OR T(2) MOD 2#0 THEN
     T(2)=200 @ GOTO 4900
```

```
4820 IF T(10)>820 OR T(10)<180 OR T(10) MOD 2#0 THEN
     T(10)=200 @ GOTO 4900
4825 IF T(11)>820 OR T(11)<180 OR T(11) MOD 2#0 THEN
     T(11)=200 @ GOTO 4900
4830 IF T(12)>820 OR T(12)<180 OR T(12) MOD 2#0 THEN
     T(12)=200 @ GOTO 4900
4835 IF T(13)>820 OR T(13)<180 OR T(13) MOD 2#0 THEN
     T(13)=200 @ GOTO 4900
4850 IF (T(2)-T(1)+T(11)-T(10)+T(13)-T(12))/2+3>10 THEN
     DISP "Zuviele Dioden " @ BEEP @ GOTO 500
4880 RETURN
4900 BEEP @ DISP "invalid Wavelengths" @ WAIT 2000 @
     GOTO 500
5000 REM ****PLOT V VS [S]*****
5005 D$="PFTN.KSYS"
5010 Y0=A(1,1) @ Y1=A(1,2) @ Y2=A(1,3) @ Y3=A(1,4)
5020 X0=0 @ X1=T(14)*1.05 @ X2=.02 @ X3=0 @ IF T(14)>.2
     THEN X2=.1
5030 IF T(14)>.4 THEN X2=.2
5035 IF T(14)>2 THEN X2=1
5040 IF T(14)>4 THEN X2=2
5042 IF T(14)>20 THEN X2=5
5044 IF T(14)>50 THEN X2=10
5046 IF T(14)>100 THEN X2=50
5048 IF T(14)>500 THEN X2=500
5050 E3$=E2$&" (uM)"
5060 E4$="Turnovernumber (1/sec)"
5090 GOSUB 6000 @ RETURN
5100 REM *****PLOT OD VS T*****
5105 D$="PFOD.KSYS"
5110 Y0=A(3,1) @ Y1=A(3,2) @ Y2=A(3,3) @ Y3=A(3,4)
5120 X0=0 @ X1=T(7) @ X2=2 @ X3=0 @ IF T(7)>15 THEN
     X2=5
5130 IF T(7)>50 THEN X2=10
```

```
5140 E3$="Time (sec)"
5150 E4$="Absorption"
5190 GOSUB 6000 @ RETURN
5200 REM **PLOT Ableit.**
5205 D$="PFAB.KSYS"
5210 Y0=A(2,1) @ Y1=A(2,2) @ Y2=A(2,3) @ Y3=A(2,4)
5220 X0=0 @ X1=T(7) @ X2=2 @ X3=0 @ IF T(7)>15 THEN
     X2=5
5230 IF T(7)>50 THEN X2=10
5240 E3$="Time (sec)"
5250 E4$="Speed (Delta OD/sec)"
5290 GOSUB 6000 @ RETURN
6000 ! PLOTTEN VON ACHSEN
6005 ON ERROR GOTO 6020 @ ASSIGN# 1 TO D$ @ READ# 1 ;
     P3,P4,X4,X5,X6,X7,Y4,Y5,Y6,Y7
6006 ASSIGN# 1 TO *
6007 IF X0#X4 OR X1#X5 OR X2#X6 OR X3#X7 THEN GOTO 6020
6008 IF Y0#Y4 OR Y1#Y5 OR Y2#Y6 OR Y3#Y7 THEN GOTO 6020
6010 RETURN
6020 ON ERROR GOTO 4000
6030 P3=2*400 @ P4=1*400 @ P5=(2+25)*400 @
     P6=(1+16)*400
6033 H1$=CHR$(3) @ P1=.000001 @ P2=.0000001
6036 F5=X1-X0 @ F6=Y1-Y0
6040 IF ABS(F5*P1)<3000 THEN P1=P1*10 @ GOTO 6040
6050 IF ABS(F6*P2)<3000 THEN P2=P2*10 @ GOTO 6050
6060 F5=(P5-P3)/100 @ F6=(P6-P4)/100
6070 ON ERROR GOTO 6090
6080 CREATE D$,1,5120
6090 ON ERROR GOTO 4000
6100 ASSIGN# 1 TO D$
6105 PRINT# 1 ; P1,P2,X0,X1,X2,X3,Y0,Y1,Y2,Y3,E3$,E4$
6110 E5$="IN;SP1;" @ GOSUB 6400
```

```
6120 E5$="IP"&VAL$(P3)&","&VAL$(P4)&","&VAL$(P5)&",
     "&VAL$(P6)&";" @ GOSUB 6400
6130 E5$="SC 0,10000,0,10000;" @ GOSUB 6400
6140 E5$="PA4000,9500;" @ GOSUB 6400 @ E5$="DT"&H1$&";"
     @ GOSUB 6400
6170 E5$="DI 1,0;" @ GOSUB 6400
6180 E5$="PU;PA 1300,1300;PD;PA 1300,9300;" @ GOSUB
     6400
6190 E5$="PA 9300,9300,9300,1300,1300,1300;PU;" @ GOSUB
     6400
6200 FOR I1=X0+X3 TO X1 STEP X2 @ I2=INT((I1-
     X0)*8000/(X1-X0)+1300)
6210 E5$="PU;PA"&VAL$(I2)&",1300;PD;XT;PU;" @ GOSUB
     6400
6220 E5$="SI.2,.25;CP-1,-1.2;LB"&VAL$(I1)&H1$&";" @
     GOSUB 6400 @ NEXT I1
6230 FOR I1=Y0+Y3 TO Y1 STEP Y2 @ I2=INT((I1-
     Y0)*8000/(Y1-Y0)+1300)
6240 E5$="PA9300,"&VAL$(I2)&";PD;YT;PU;" @ GOSUB 6400 @
     NEXT I1
6250 FOR I1=Y0+Y3 TO Y1 STEP Y2 @ I2=INT((I1-
     Y0)*8000/(Y1-Y0)+1300)
6260 E5$="PA1300,"&VAL$(I2)&";PD;YT;PU;" @ GOSUB 6400
6270 E5$="CP-5,-.5;LB"&VAL$(I1)&H1$&";" @ GOSUB 6400 @
     NEXT I1
6280 FOR I1=X0+X3 TO X1 STEP X2 @ I2=INT((I1-
     X0)*8000/(X1-X0)+1300)
6290 E5$="PU;PA"&VAL$(I2)&",9300;PD;XT;PU;" @ GOSUB
     6400 @ NEXT I1
6300 P3=INT(P3+F5*13)-50 @ P4=INT(P4+F6*13) @
     P5=INT(P5-F5*7)-470 @ P6=INT(P6-F6*7)
6310 E5$="IP"&VAL$(P3)&","&VAL$(P4)&","&VAL$(P5)&",
     "&VAL$(P6)&";" @ GOSUB 6400
```

```
6320 E5$="IW"&VAL$(P3)&","&VAL$(P4)&","&VAL$(P5)&",
     "&VAL$(P6)&";" @ GOSUB 6400
6330 H1=INT(X0*P1) @ H2=INT(X1*P1) @ H3=INT(Y0*P2) @
     H4=INT(Y1*P2)
6340 E5$="SC"&VAL$(H1)&","&VAL$(H2)&","&VAL$(H3)&",
     "&VAL$(H4)&";" @ GOSUB 6400
6350 ASSIGN# 1 TO *
6360 SECURE D$,"XX",3 @ RETURN
6390 !  TO DISK **
6400 PRINT# 1 ; CHR$(34),E5$,CHR$(34) @ RETURN
9000 ! **error recovery ***
9010 BEEP @ BEEP @ BEEP
9100 IF ERRN=1 THEN DISP "too small number" @ GOTO 9500
9110 IF ERRN=2 THEN DISP "too large number" @ GOTO 9500
9120 IF ERRN=7 THEN DISP "Null Data" @ GOTO 9500
9130 IF ERRN=8 THEN DISP "Division by zero" @ GOTO 9500
9140 IF ERRN=11 THEN DISP "Argument out of Range" @
     GOTO 9500
9150 IF ERRN=22 THEN DISP "File is secured" @ GOTO 9500
9155 IF ERRN=43 THEN DISP "Numeric input " @ GOTO 9500
9160 IF ERRN=44 THEN DISP "not enough inputs" @ GOTO
     9500
9170 IF ERRN=45 THEN DISP "too many inputs" @ GOTO 9500
9180 IF ERRN=49 THEN DISP "Null Data" @ GOTO 9500
9190 IF ERRN=56 THEN DISP "String Overflow" @ GOTO 9500
9200 IF ERRN=55 THEN DISP "String Subscript" @ GOTO
     9500
9210 IF ERRN=67 THEN DISP "Filename doesn't exist" @
     GOTO 9500
9215 IF ERRN=68 THEN DISP "Filetyp wrong" @ GOTO 9500
9220 IF ERRN=89 THEN DISP "invalid parameter" @ GOTO
     9500
9230 IF ERRN=108 THEN DISP "Photometer Waring" @ GOTO
     9500
```

```
9240 IF ERRN=111 THEN DISP "No disk inserted" @ GOTO
     9500
9250 IF ERRN=125 THEN DISP "Volume not found" @ GOTO
     9500
9260 IF ERRN=128 THEN DISP "Disk is full" @ GOTO 9500
9265 IF ERRN=130 THEN DISP "no disk" @ GOTO 9500
9270 IF ERRN=131 THEN DISP "Time Out" @ GOTO 9500
9490 DISP "Error ";ERRN;" ocurred on line ";ERRL @
     PAUSE
9495 GOTO 4000
9500 WAIT 3000 @ GOTO 4000
```

APPENDIX H

Measure Overlay Kin 2

```
1 ! KIN 2 B.MICHEL
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
     D3$[20],D4$[20],B9$[8]
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],F5$[1],
     A$[60],H1$[20]
195 INTEGER I,I2,I5,I6
200 F6$="0" @ Q1=1 @ Q2=1 @ F7$="0" @ CLEAR @ I=0
210 F$="A" @ F5$="0" @ J=1 @ K=1 @ D4$[5,5]="1"
250 ON ERROR GOTO 800 @ PLOTTER 1 @ IF C(8)#1 THEN GOTO
     800
500 L1=B(11) @ L2=B(12) @ L3=256^(1/T(5)) @ ON ERROR
     GOTO 500
510 CLEAR @ DISP "    Km Measure : Main Menu" @ DISP
520 DISP E1$;TAB(19);T(3);"nm (E)" @ DISP
     E2$;TAB(19);T(4);"m (S)" @ DISP
530 DISP "No. of Rep.";T(8) @ DISP "No. of Points
     ";T(5)
650 ON KEY# 1,"DiVar" GOTO 720 @ ON KEY# 2,"MeasM" GOTO
     1000 @ ON KEY# 3,"Syringe" GOTO 860
655 ON KEY# 4,"Wash" GOTO 700 @ ON KEY# 5,"MAINPG" GOTO
     800 @ ON KEY# 6,"TestM" GOTO 790
660 ON KEY# 7,"[E,S]" GOTO 880 @ ON KEY# 8,"Points"
     GOTO 940 @ KEY LABEL @ GOTO 650
700 GOSUB 7500 @ GOTO 500
720 PRINTER IS 1 @ GOSUB 9545 @ PAUSE
725 GOTO 500
760 CLEAR @ IF B(4)=0 THEN B(4)=1 @ DISP "Enter v in
     assay";@ INPUT B(10)@ GOTO 765
```

```
762 B(4)=0 @ DISP "Enter [E] in Assay";@ INPUT B(5)
765 B(3)=B(5) @ GOTO 1000
790 RESET 7 @ CHAIN "KONZMSG.KSYS"
800 RESET 7 @ CHAIN "Autost.KSYS"
820 CLEAR @ DISP "** Single Meas **"
825 DISP E2$ @ DISP "Conc in M at [S]<0: Reference";
830 INPUT B(6)@ C(10)=1
835 DISP E1$ @ DISP "Conc in nM (Stock=";T(3);"nM)";@
     INPUT B(5)
840 B(3)=B(5) @ MODE 0,1 @ GOSUB 4800 @ GOTO 1000
860 GOSUB 7250 @ GOTO 500
880 CLEAR @ DISP E1$;" Conc. in (nM)"
885 DISP "and ";E2$;" Conc. in (M)" @ INPUT T(3),T(4)
890 GOTO 500
900 CLEAR @ IF B(20)=2 THEN DISP E1$;ELSE DISP E2$;
905 DISP " Conc. Nr.";C(6)
910 DISP "in Syringe (C=clear Buffer)" @ DISP "negative
     [E] = decreasing [S]" @ INPUT H1$
915 IF H1$="C" THEN C(6)=1 @ GOTO 900
920 E(C(6))=VAL(H1$) @ C(6)=C(6)+1 @ RETURN
940 CLEAR @ DISP "No. of Points (2-40) and Rep. (2-
     10)";
950 INPUT T(5),T(8)@ IF T(5)>40 OR T(5)<2 OR T(8)>10 OR
     T(8)<0 THEN GOTO 940 ELSE GOTO 500
1000 CLEAR @ DISP "     Km Measure Menu" @ DISP @ DISP
     T1$[1,63] @ DISP @ ON ERROR GOTO 500
1020 DISP "Syr.1:";INT(C(1)*10)/10;"2:";INT(C(2)*
     100)/100;"3:";INT(C(3)*100)/100;"ml" @ DISP
1050 DISP "Max. [S]  :  ";T(14);"uM"
1090 IF D4$[4,4]="L" THEN DISP "Blank ";D4$[6,14];"
     subtracted" ELSE DISP "No Blank"
1120 IF B(4)=1 THEN DISP "[E] Autom."
1130 IF D4$[5,5]="1" THEN DISP "Print On" ELSE DISP
     "Print OFF"
```

```
1140 IF C(9)=0 THEN DISP "No Reference" @ BEEP
1150 DISP "Assay Volume : ";B(7);"ul" @ ON ERROR GOTO
     1000
1200 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"Print" GOTO
1280 @ ON KEY# 3,"Content" GOTO 1600
1230 ON KEY# 4,"Comment" GOTO 1690 @ ON KEY# 5,"KmMeas"
     GOTO 1300 @ ON KEY# 6,"Blank" GOTO 1500
1260 ON KEY# 7,"[E]auto" GOTO 760 @ ON KEY#
     8,"SingleMe" GOTO 820 @ KEY LABEL @ GOTO 1200
1280 IF D4$[5,5]="0" THEN D4$[5,5]="1" ELSE
     D4$[5,5]="0"
1290 GOTO 1000
1300 ON ERROR GOTO 1000 @ IF C(9)=0 THEN GOTO 1000
1302 CLEAR @ DISP "Series and Number eg A,1";@ INPUT
     D1$,B1@ IF B1>999 OR B1<0 THEN 1300
1304 DISP "Memocode (6 Characters)";@ INPUT D4$[15,20]@
     A$="*4" @ GOSUB 7000
1310 M=B(20) @ IF C(M)<W(M)/8 THEN H1=7 @ GOSUB 7510
1320 D2=B1 @ B1=B1+1 @ F8$="N"
1325 IF C(6)=1 THEN GOSUB 900 ELSE H1=0 @ PRINTER IS 1
     @ GOSUB 9500
1330 B(3)=B(5) @ IF E(1)#0 THEN Q1=1 @ GOTO 1335
1332 E(1)=1 @ B(3)=.1 @ Q1=2 @ D4$[4,4]="0" @
     D1$=CHR$(NUM(D1$)+1) @ D2=0
1335 IF E(1)<0 THEN Q2=2 @ E(1)=-E(1) ELSE Q2=1
1337 IF B(20)=2 THEN T(3)=E(1) ELSE T(4)=E(1)
1340 FOR I=1 TO C(6)-1 @ E(I)=E(I+1) @ NEXT I @
     C(6)=C(6)-1
1350 W1(2)=B(7)*B(5)/T(3) @ GOSUB 8000 @ GOTO 2000
1380 IF C(6)<2 THEN 1000 ELSE CLEAR @ GOTO 1320
1500 IF D4$[4,4]="L" THEN D4$[4,4]="0" @ GOTO 1000
1510 CLEAR @ DISP "Name of Blankfile (9 Characters)";@
     INPUT H1$@ D4$[4,4]="L"
1515 IF LEN(H1$)<9 THEN GOTO 1510
```

```
1520 H1$=H1$&"    " @ D4$[6,14]=H1$
1540 CREATE D4$[6,14]&".KSYS",5,256 @ ASSIGN# 1 TO
     D4$[6,14]&".KSYS"
1560 FOR J=1 TO 20 @ PRINT# 1 ; T(J) @ NEXT J
1570 FOR J=0 TO T(5) @ PRINT# 1 ; T1(J),T2(J) @ NEXT J
1580 ASSIGN# 1 TO * @ H1$=D4$[6,14]
1590 PRINTER IS 2 @ PRINT "Blank Stored in File: ";H1$
     @ RETURN
1600 CAT "."&B9$ @ DISP "Purge File" @ INPUT H1$@ PURGE
     H1$ @ GOTO 1000
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP "Correct
     Error condition  +CONT";ERRL
1670 PAUSE
1675 GOTO 500
1690 GOSUB 1700 @ GOTO 1000
1700 DISP "Comment 2 Lines" @ FLIP @ INPUT T1$@ FLIP
1710 T1$=T1$&" " @ IF LEN(T1$)<63 THEN GOTO 1710
1720 RETURN
1800 RESET 7 @ ASSIGN# 1 TO D4$[6,14]&".KSYS"
1820 FOR J=1 TO 20 @ READ# 1 ; Z8@ IF J=3 OR J=4 OR J=6
     OR J=7 OR J=15 OR J=16 THEN GOTO 1840
1835 IF Z8#T(J) THEN BEEP @ GOTO 1885
1840 NEXT J @ Z8=1
1850 FOR K=0 TO T(5)
1860 READ# 1 ; T1(K),T2(K)
1870 T1(K)=-T1(K) @ T2(K)=-T2(K) @ NEXT K @ ASSIGN# 1
     TO *
1875 H1$=D4$[6,14] @ PRINTER IS 2 @ PRINT "Blank:
     ";H1$;" subtracted" @ RETURN
1885 DISP "Variable";J;T(J);"# Blank:";Z8
1890 DISP "Change Variable or (999=ignore)";@ INPUT H1@
     IF H1=999 THEN 1840
1892 IF J=1 OR J=2 OR J=10 OR J=11 OR J=12 OR J=13 OR
     J=19 THEN BEEP @ GOTO 1000 ELSE T(J)=H1
```

```
1895 GOTO 1840
1900 A$="*1" @ GOSUB 7000 @ W(11)=.5 @ A$="*A0.5" @
     GOSUB 7000
1905 A$="*B" @ GOSUB 7000 @ WAIT 2000 @ GOSUB 7000 @
     WAIT 2000
1910 MODE 0,1
1920 REFERENCE 10
1930 IF NMEAS=0 THEN GOTO 1930
1940 MEASURE .1
1950 IF NMEAS=0 THEN GOTO 1950
1960 FOR J=1 TO 10 @ IF VALUE(L(J))>5 THEN GOTO 1900
1970 NEXT J @ C(9)=1 @ RETURN
2000 ERASE STATUS
2020 IF T(14)<T(4)/20 OR T(14)>T(4)/3 THEN BEEP @ DISP
     "[S] wrong !" @ WAIT 3000 @ GOTO 500
2040 GOSUB 3600 @ CLEAR @ PRINTER IS 1 @ H1=0 @ GOSUB
     9510
2050 DISP "1=Exit 2=Input"
2200 GOSUB 3650 @ Z5=0
2230 F7$="0" ! TP
2240 IF Q2=2 THEN GOTO 2247
2245 FOR K=0 TO T(5) @ GOTO 2249
2247 FOR K=T(5) TO 0 STEP -1
2249 Z5=Z5-1 @ IF Z5<0 THEN Z5=0
2250 IF F7$="2" THEN F7$="0" @ K=P3
2255 FOR J=1 TO 80 @ S3(J)=0 @ NEXT J
2260 R9=0 @ R8=0 @ S6=0 @ S7=0
2300 T(0)=INT(T(20)-K*15/T(5)) @ IF K=0 OR T(0)<0 OR
     T(0)>T(20) THEN T(0)=0
2302 T(0)=T(0)+T(8) @ IF T(0)>10 THEN T(0)=10
2305 FOR J=1 TO T(0) @ F6$="0"
2310 GRAPH
2350 GOSUB 3500 ! MESSE
2420 GOSUB 4600 ! DERIV
```

```
2430 GOSUB 3300 ! EM TEST
2450 IF F5$="1" THEN GOTO 2310
2470 FOR J1=2 TO T9/L2+1
2480 S3(J1)=S3(J1)+S1(J1)
2490 NEXT J1 @ R9=R9+S7 @ R8=R8+S6 @ IF F6$="1" THEN
     GOTO 2530
2500 NEXT J
2530 GOSUB 3700 ! MITT
2580 IF F6$="1" THEN GOTO 2595
2590 FOR J=2 TO T9/L2+1 @ S3(J)=S3(J)/T(0) @ NEXT J @
     IF F6$="2" THEN GOSUB 4510
2595 F6=3 @ GOSUB 6500
2600 GOSUB 3800 ! CHECK
2620 IF F5$="1" THEN GOSUB 4500
2650 T1(K)=T1(K)+T1(41) @ T2(K)=T2(K)+T2(41) @ IF
     D4$[5,5]="0" THEN 2654
2651 IF ABS(T1(K))>1 OR ABS(T2(K))>1 OR ABS(S9)>1 THEN
     PRINT K;T1(K);T2(K);S9 @ GOTO 2654
2652 PRINTER IS 2 @ PRINT USING "DD,X,SD.6D,X,SD.6D,X,
     D.6D" ; K,T1(K),T2(K),S9
2654 IF F7$="1" THEN F7$="0" @ GOTO 3000
2655 NEXT K
2995 DISP T1$ @ DISP @ DISP @ A$="*WFCC8" @ GOSUB 7000
     @ GOSUB 7700 @ I=BIT(I5,2)
2997 IF C(6)>1 AND I=0 THEN A$="*5" @ GOSUB 7000 @ WAIT
     1000 @ A$="*I" @ GOSUB 7000
3000 ON KEY# 3,"Rep.Me" GOTO 3100 @ ON KEY# 2,"Output"
     GOTO 3250 @ ON ERROR GOTO 3000
3010 ON KEY# 8,"Syringe" GOTO 3050 @ ON KEY# 4,"Next
     Pr" GOTO 3030 @ ON KEY# 1,"Exit" GOTO 500
3015 ON KEY# 5,"ResM" GOTO 3060 @ ON KEY# 6,"** End-M"
     GOTO 3000 @ ON KEY# 7,"enu **" GOTO 3000
3017 KEY LABEL @ J=0
3019 I5=SPOLL(708) @ I=BIT(I5,3) @ J=J+1 @ BEEP 300,10
```

```
3020 WAIT 2000 @ IF C(6)>1 AND J>2 AND I=1 THEN 3250
     ELSE 3019
3030 GOSUB 900 @ J=2 @ GOTO 3019
3050 GOSUB 7250 @ GOTO 3000
3060 F7$="2" @ CLEAR @ DISP "Proceed from Point No." @
     INPUT K@ P3=K @ GOSUB 3110
3070 F5$="0" @ GOTO 2240
3100 F7$="1" @ CLEAR @ DISP "Repeat which Point" @
     INPUT K@ GOSUB 3110 @ GOTO 2255
3110 IF D4$[4,4]#"L" THEN T1(K)=0 @ T2(K)=0 @ GOTO 3200
3120 ASSIGN# 1 TO D4$[6,14]&".KSYS" @ FOR J=1 TO 20 @
     READ# 1 ; T1(41)@ NEXT J
3130 FOR J=0 TO T(5) @ READ# 1 ; Y0,Y1@ IF K=J AND
     F7$="1" THEN T1(K)=-Y0 @ T2(K)=-Y1
3140 IF F7$="2" AND Q2=2 AND K>=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3145 IF F7$="2" AND Q2=1 AND K<=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3150 NEXT J @ ASSIGN# 1 TO *
3200 RETURN
3250 IF F$#"A" OR C(6)<2 THEN 3255
3251 A$="*4" @ GOSUB 7000 @ I=B(20) @ IF E(1)=0 THEN
     H1=W(I+5) @ GOTO 3253
3252 H1=ABS(W(I+5)/E(1)) @ IF I=2 THEN H1=H1*T(3) ELSE
     H1=H1*T(4)
3253 A$="*"&VAL$(I) @ GOSUB 7000 @ W(I+5)=H1 @
     A$="*="&VAL$(H1) @ GOSUB 7000
3254 A$="*E8" @ GOSUB 7000
3255 IF Q1=2 THEN RESET 7 @ H1$=D4$[15,20]&D1$&VAL$
     (D2)&"N" @ GOSUB 1520 @ D4$[4,4]="L"
3270 GOSUB 4200 @ GOTO 1380
3300 I3=0 @ S7=0 @ F5$="0" @ ON KEY# 2 GOTO 3390 @ BEEP
     100,10 ! EMTES
```

```
3310 FOR J1=T8/L2+1 TO T9/L2+1 @ H1=(S9-S1(J1))^2 @
     I3=I3+1 @ S7=S7+H1
3320 NEXT J1 @ S6=(S7/I3)^.5 @ IF S6<.00001*(1+K/T(5))
     *B(2) THEN GOTO 3350
3330 IF K=0 THEN GOTO 3350
3340 IF S6>ABS(S9/100*B(2)) THEN DISP "Meas";K;J;
     "Repeated";Z5 @ BEEP @ F5$="1" @ Z5=Z5+1
3345 IF Z5>3*T(8) AND F5$="1" THEN PRINT J;S9;S8;" ??"
     @ F5$="0"
3350 BEEP 200,10 @ OFF KEY# 2 @ RETURN
3390 I6=1 @ GOSUB 4450 @ GOSUB 4550 @ GOTO 3300
3500 T8=INT((B(13)+K/T(5)*(T(6)-B(13)))/L2)*L2
3503 T9=INT((B(14)+K/T(5)*(T(7)-B(14)))/L2)*L2
3505 IF J#1 THEN GOTO 3510 ELSE I=0 @ GOSUB 7900 @ WAIT
     500
3507 IF K<2 OR K=T(5) OR T(5)<7 THEN A$="*H" @ GOSUB
     7000 @ WAIT 3000+B(9)*K/T(5)
3510 MODE 0,1 @ A$="*H" @ GOSUB 7000 @ WAIT B(8)+B(9)
     *K/T(5)
3512 H4=(T(11)-T(10))/2+1 @ H2=(T(2)-T(1))/2+1 @
     H3=(T(13)-T(12))/2+1
3514 ERASE MEMORY -1 ! MESSE
3515 ON ERROR GOTO 3597
3520 MEASURE L1,L2,0,T9+L2
3530 FOR J1=I TO T9/L2+2
3540 IF NMEAS#J1 THEN GOTO 3540
3550 TO MEMORY J1
3560 IF L1>.2 THEN GOSUB 4400
3570 NEXT J1 @ I6=SPOLL(708) @ IF BIT(I6,0)=1 THEN 2995
3575 ALPHA @ IF L2>.2 THEN 3595
3580 FOR J1=1 TO T9/L2+2
3585 RECALL MEMORY J1
3587 GOSUB 4400 @ NEXT J1
3595 RETURN
```

```
3597 IF ERRN=2 THEN GOSUB 1900 @ PRINT "Ref. repeated"
     @ GOTO 3500 ELSE GOTO 1660
3600 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     3610
3605 ASSIGN# 1 TO D$ @ DISP "File exists" @ ASSIGN# 1
     TO * @ BEEP @ WAIT 2000 @ GOTO 1000
3610 IF ERRN=130 OR ERRN=125 THEN DISP B9$;"not found"
     @ BEEP @ WAIT 2000 @ GOTO 3600
3620 ON ERROR GOTO 1660 @ PLOTTER 1 @ PRINTER 2
3625 K=T(5) @ GOSUB 8250 @ L(20)=L(20)*T(14)/H4 @
     C(10)=0
3630 B(6)=T(4)/20 @ GOSUB 4800 @ GOSUB 8000 @ IF
     D4$[5,5]="0" THEN H1=0 ELSE H1=1
3635 GOSUB 9500
3640 FOR K=0 TO 40 @ T1(K)=0 @ T2(K)=0 @ NEXT K
3645 IF D4$[4,4]="L" THEN GOSUB 1800
3647 RETURN
3650 F6$="0" @ IF D4$[5,5]="1" THEN PRINTER 2 @ PRINT @
     PRINT "NR   Range 1   Range 2  Noise 1"
3655 DISP "Duration";INT(T(8)*T(5)*(T(7)+5)/200+1)
     *3;"Minutes"
3660 DISP "No.      Range 1     Range 2"
3680 RETURN
3700 T1(41)=0 @ T2(41)=0 @ H4=W1(3)*T(4)/B(7)
3710 FOR J=1 TO T(0) @ T1(41)=T1(41)+S4(J) @
     T2(41)=T2(41)+S5(J) @ NEXT J
3720 T1(41)=T1(41)/T(0) @ T2(41)=T2(41)/T(0) @
     R9=R9/T(0) @ R9=R9^.5 @ R8=(R8/T(0))^.5
3730 DISP USING "DD,X,5D.DD,XXX,DDD.DD,XXX,.6D" ;
     K,H4,B(3),R8 @ RETURN
3800 S9=0 @ S8=0 @ F5$="0" ! CH
3810 FOR J=1 TO T(0)
3820 S9=S9+(T1(41)-S4(J))^2 @ S8=S8+(T2(41)-S5(J))^2 @
     NEXT J @ S9=(S9/T(0))^.5
```

```
3830 IF S9<ABS(.00005*B(1)) OR Q1#1 THEN GOTO 3860
3850 IF S9>ABS(T1(41)/100*B(1)) THEN DISP "Average";K;"
     repeated";Z5 @ BEEP @ F5$="1"
3855 Z5=Z5+T(0) @ IF Z5>2*T(8) AND F5$="1" THEN 3870
3860 RETURN
3870 PRINT "Average nonreliable" @ FOR J=1 TO T(0) @
     PRINT "Msg";K;J;" ";S4(J);S5(J) @ NEXT J
3875 F5$="0" @ GOTO 3860
4200 IF D4$[5,5]="0" THEN 4205
4201 PRINTER 2 @ PRINT @ PRINT @ PRINT TAB(7);E4$ @
     GRAPH @ COPY
4202 H1=0 @ GOSUB 9500 @ PRINT T1$ @ ON ERROR GOTO 1660
4205 PRINT @ PRINT @ PRINT "No.  Range 1   Range 2
     [Substr]"
4210 PRINT "   TN(1/sec) TN(1/sec)    M" @ PRINT
4220 K=T(5) @ GOSUB 8250
4230 A$="PFTN.KSYS"
4250 GOSUB 8500
4260 LORG 0 @ CSIZE 3 @ GOSUB 5500
4270 FOR K=0 TO T(5) @ GOSUB 8250
4290 MOVE H4,H3 @ LABEL "*" @ MOVE H4,J1 @ LABEL "+"
4310 PRINTER IS 2 @ PRINT USING "DD,X,S5D.D,X,
     S5D.D,XX,5D.DD" ; K,H3,J1,H4
4312 PRINT# 1 ; H3,J1,H4
4320 NEXT K @ PRINTER IS 2
4330 PRINT @ IF D4$[5,5]="1" THEN PRINT @ PRINT
     TAB(7);E4$ @ COPY @ PRINT @ PRINT
4350 ASSIGN# 1 TO *
4360 GOSUB 6000 @ PRINT @ PRINT @ PRINT @ RETURN
4400 I3=0 @ H1=0
4410 FOR I2=T(1) TO T(2) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4415 S1(J1)=H1/H2 @ H1=0
```

```
4420 FOR I2=T(12) TO T(13) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4425 S2(J1)=H1/H3 @ H1=0
4430 FOR I2=T(10) TO T(11) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4435 S1(J1)=S1(J1)-H1/H4 @ S2(J1)=S2(J1)-H1/H4
4440 RETURN
4450 DISP "1=Exit 2=New Stdev 3=Rep.Single" @ DISP
     "4=Rep.Aver. 5=End Me  6=go on"
4455 DISP "7=Test Msg 8=Next Ready";
4460 INPUT H1@ RETURN
4500 IF F$="A" THEN GOTO 2250
4510 GOSUB 4450 @ ON ERROR GOTO 1660 @ I6=2
4520 F6$="0" @ IF H1#3 THEN 4550
4530 F6$="1" @ DISP "No. of Single Measurement" @ INPUT
     J@ FOR J1=1 TO 80 @ S3(J1)=0 @ NEXT J1
4540 R9=R9^2*T(0)/(1.2+T(0)) @ R8=R8^2*T(0)/(1.2+T(0))
     @ GOTO 2310
4550 IF H1=5 THEN 2995
4555 IF H1=7 THEN 2000
4560 IF H1=2 THEN DISP "Max. Nonlin./Noise 2 Inputs";@
     INPUT B(1),B(2)@ GOTO 4590
4565 IF H1=8 THEN GOSUB 900 @ GOTO 4590
4570 IF H1=3 AND I6=1 THEN F6$="2" @ GOTO 4590
4580 IF H1=4 THEN 2250
4590 RETURN
4600 S9=0 @ S8=0 @ I1=0
4620 H1=S1(1) @ H2=S2(1)
4630 FOR J1=2 TO T9/L2+1
4635 H3=S1(J1) @ H4=S2(J1)
4640 S1(J1)=(S1(J1+1)-H1)/(2*L2) @ S2(J1)=(S2(J1+1)-
     H2)/(2*L2) @ IF J1>T9/L2+1 THEN GOTO 4710
4700 IF J1>=T8/L2+1 THEN I1=I1+1 @ S9=S9+S1(J1) @
     S8=S8+S2(J1)
```

```
4710 H1=H3 @ H2=H4 @ NEXT J1 @ I3=0 @ S8=S8/I1 @
     S9=S9/I1
4730 DISP USING "DD,X,DD,XX,SZ.6D,XXX,SZ.6D" ;
     K,J,S9,S8
4740 S4(J)=S9 @ S5(J)=S8
4750 RETURN
4800 F6$="0" @ A5=0 @ A6=0 @ MODE 0,1
4810 LAMBDA L(1),L(2),L(3),L(4),L(5),L(6),L(7),L(8),
     L(9),L(10)
4820 TIME SCALE 0 TO T(7)
4830 ABSORBANCE
4832 IF B(3)>T(3)/3 THEN B(3)=T(3)/3
4833 IF B(6)>T(4)/3 THEN B(6)=T(4)/3
4835 W1(3)=B(7)*B(6)/T(4)
4836 W1(2)=B(7)*B(3)/T(3)
4840 IF W1(3)<=0 THEN GOTO 1900
4845 IF C(9)#1 THEN 1000
4850 I=0 @ GOSUB 7980
4860 A$="*H" @ GOSUB 7000
4865 WAIT 5000
4870 J=0 @ K=INT(T(5)/2) @ T9=T(7) @ T8=B(13) @ GOSUB
     3510
4920 H1=0 @ H2=0
4930 FOR J1=1 TO 4 @ H1=S1(J1)+H1 @ H2=S2(J1)+H2 @ NEXT
     J1 @ H1=H1/4 @ H2=H2/4
5050 IF D4$[5,5]#"0" AND A5#1 THEN A$="PFOD.KSYS" @
     GOSUB 8500 @ A5=1
5055 DISP "Initial OD Range 1 and Range 2"
5056 DISP USING "6X,SD.5D,6X,SD.5D" ; H1,H2 @
     H1=H1/T(15)*1000 @ H2=H2/T(16)*1000
5057 IF ABS(H1)>99999 OR ABS(H2)>99999 THEN DISP
     "[E]";H1;H2;"uM" @ GOTO 5060
5058 DISP USING "3A,X,S5D.3D,4X,S5D.3D,2A" ; "[E]",
     H1,H2,"uM" @ DISP "Deriv. Range1 and Range 2"
```

```
5060 IF D4$[5,5]#"0" THEN F6=1 @ GOSUB 6500 @ F6=2 @
     GOSUB 6500
5150 K=0 @ J=0 @ GOSUB 4600 @ GOSUB 3300 @
     H4=ABS(S6/S9*100) @ IF H4>999 THEN H4=999
5160 DISP USING "14A,X,.6D,3A,3D.D,A" ; "Noise  Range
     1",ABS(S6)," = ",H4,"%" @ H1=0 @ H2=0
5170 FOR J=2 TO 6 @ H1=H1+S1(J) @ NEXT J @ H1=H1/5 @
     A1=H4
5180 FOR J=T(7)/L2-3 TO T(7)/L2+1 @ H2=H2+S1(J) @ NEXT
     J @ H2=H2/5 @ H3=INT(ABS((H1-H2)/H1*100))
5190 DISP "Nonlinearity  : ";H3;"%" @ A2=H3 @ DISP @
     DISP
5300 J1=0 @ IF D4$[5,5]#"0" THEN GOSUB 5400
5310 IF J1=9 THEN 4810
5350 IF A5#2 THEN A$="PFAB.KSYS" @ GOSUB 8500 @ A5=2
5360 F6=1 @ GOSUB 6500 @ F6=2 @ GOSUB 6500
5390 GOSUB 5400 @ IF J1=9 THEN 4810
5395 RETURN
5400 OFF KEY# 8 @ ON KEY# 2,"Go on" GOTO 5490 @ ALPHA
5402 ON KEY# 5,"[E]" GOTO 5440 @ ON KEY# 6,"Activ" GOTO
     5445 @ OFF KEY# 7
5405 ON KEY# 3,"Graph" GOTO 5420 @ ON KEY# 4,"Alpha"
     GOTO 5430 @ KEY LABEL
5407 IF F$="A" AND C(10)=0 THEN J1=1 ELSE J1=0
5410 IF J1=0 THEN 5410 ELSE BEEP 300,20 @ WAIT 3000 @
     GOTO 5450
5420 GRAPH @ J1=0 @ GOTO 5410
5430 ALPHA @ J1=0 @ GOTO 5410
5440 DISP B(3);"new";@ INPUT B(3)@ B(5)=B(3) @ J1=9 @
     GOTO 5490
5445 DISP B(10);"new";@ INPUT B(10)@ GOTO 5400
5450 IF B(4)=0 OR Q1=2 THEN 5490
5454 IF B(3)<T(3)/200 THEN B(3)=T(3)/200 @ GOTO 5490
5455 IF ABS(S9)>2*B(10) THEN H1=.5 @ GOTO 5492
```

```
5460 IF ABS(S9)<B(10)/2 THEN H1=2 @ GOTO 5492
5465 IF ABS(S9)>1.3*B(10) THEN H1=.75 @ GOTO 5492
5470 IF ABS(S9)<.75*B(10) THEN H1=1.25 @ GOTO 5492
5490 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ RETURN
5492 B(3)=B(3)*H1 @ A6=A6+1 @ CLEAR @ DISP
     "Enzymeconc.:";B(3) @ IF A6>6 THEN J1=0 ELSE J1=9
5493 GOTO 5490
5495 GOSUB 7250 @ GOTO 5400
5500 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     1660
5550 CREATE D$,3+INT(T(5)/10+.5),256
5560 ASSIGN# 1 TO D$
5570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B()
5660 RETURN
6000 A1=0 @ A2=0 @ A3=0 @ A4=0 @ A5=0 @ A6=0
6050 FOR K=2 TO T(5) @ GOSUB 8250
6090 IF H3=0 OR H4=0 THEN GOTO 6200 ELSE H1=1/H4 @
     H2=1/H3
6120 A1=A1+1 @ A2=A2+H1 @ A3=A3+H2 @ A4=A4+H1*H2 @
     A5=A5+H1*H1 @ A6=A6+H2*H2
6200 NEXT K
6210 IF A1=0 THEN GOTO 6400
6220 H1=A5/A1-(A2/A1)^2
6230 H2=A6/A1-(A3/A1)^2
6240 H3=(A4/A1-A2/A1*A3/A1)/H1
6250 H4=A3/A1-H3*A2/A1
6260 H5=H3*H1^.5/H2^.5
6280 H1=ABS(1/(H4/H3)) @ H2=ABS(1/H4)
6290 IF ABS(H1)>9999 OR ABS(H2)>99999 OR ABS(H5)>1 THEN
     GOTO 6400
6300 PRINT USING "15A,4X,6D.2D,X,3A" ; "Velocity
     =",H2,"1/s"
```

```
6310 PRINT USING "16A,3X,5D.3D,X,3A" ; "Michaelis
     Const.=",H1," M"
6320 PRINT USING "14A,9X,3D.4D" ; "Correlation  =",H5
6400 PRINT @ PRINT @ RETURN
6500 PLOT 0,0,2
6520 FOR J1=2 TO T9/L2+1
6530 H4=(J1-1)*L2
6540 ON F6 GOTO 6550,6560,6570
6550 H3=S1(J1) @ GOTO 6580
6560 H3=S2(J1) @ GOTO 6580
6570 H3=S3(J1)
6580 PLOT H4,H3,1
6590 NEXT J1 @ PENUP @ PLOT 0,0,2 @ RETURN
7000 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
7005 SET TIMEOUT 7;3000
7010 ON TIMEOUT 7 GOTO 7090
7015 A1$=A$[2,2]
7020 I5=SPOLL(708)
7025 IF BIT(I5,7) THEN WAIT 500 @ GOTO 7020
7030 IF BIT(I5,5) THEN DISP "Empty" @ F5$="2" @ C(15)=0
     @ GOTO 7082
7032 IF BIT(I5,4) THEN DISP "Full" @ C(15)=0
7035 IF BIT(I5,3) THEN C(15)=0
7040 A$=A$&"," @ OUTPUT 708 ;A$
7052 IF A1$="1" THEN M=1
7054 IF A1$="2" THEN M=2
7060 IF A1$="3" THEN M=3
7061 IF A1$="B" OR A1$="D" THEN C(M)=C(M)-W(M+10)
7062 IF A1$="C" THEN C(M)=C(M)+W(M+10)
7063 IF A1$="F" THEN C(M)=0
7064 IF A1$="G" THEN C(M)=W(M)
7065 IF A1$="H" THEN C(1)=C(1)-W1(1) @ C(2)=C(2)-W1(2)
     @ C(3)=C(3)-W1(3)
```

```
7070 OFF TIMEOUT 7
7080 RETURN
7082 IF A1$="F" OR A1$="B" OR A1$="D" THEN 7080 ELSE
7040
7090 DISP "Switch on ASSAYOMAT" @ BEEP @ WAIT 5000 @
     RESET 7 @ GOTO 7000
7100 DISP A$ @ A$="*TF84A " @ I6=W1(4) @ GOSUB 7200
7110 I6=INT(W1(4)/(W1(1)*48000/W(1)))+257 @ GOSUB 7200
7115 IF W1(2)=0 THEN I6=60258 ELSE I6=INT(W1(4)/
     (W1(2)*48000/W(2)))+257
7120 GOSUB 7200 @ IF W1(3)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(3)*48000/W(3)))+257
7125 GOSUB 7200
7130 GOSUB 7000
7190 RETURN
7200 B$="     " @ H1$="0123456789ABCDEF"
7205 FOR J1=4 TO 1 STEP -1
7210 I=I6 MOD 16 @ B$[J1,J1]=H1$[I+1,I+1] @ I6=I6-I @
     I6=I6/16 @ NEXT J1
7215 A$=A$&"="&B$[3,4]&"="&B$[1,2]
7220 RETURN
7250 M=0 @ A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700 @ GOTO
     7480
7280 ON KEY# 1,"Give" GOTO 7430 @ ON KEY# 2,"Suck" GOTO
     7420
7300 ON KEY# 3,"Back" GOTO 7400 @ ON KEY# 4,"Empty"
     GOTO 7460
7320 ON KEY# 5,VAL$(W(M+10)) GOTO 7360
7330 ON KEY# 6,VAL$(C(M)) GOTO 7440
7340 ON KEY# 7,"Retur" GOTO 7355
7350 ON KEY# 8,"Syr."&VAL$(M) GOTO 7480
7352 KEY LABEL @ GOTO 7280
7355 OFF KEY# 1 @ RETURN
```

```
7360 CLEAR @ DISP "Volume per Key";@ INPUT H1$@
     W(M+10)=VAL(H1$) @ A$="*A"&H1$ @ GOSUB 7000
7370 GOTO 7280
7400 A$="*D" @ GOSUB 7000 @ GOTO 7280
7420 A$="*C" @ GOSUB 7000 @ GOTO 7280
7430 A$="*B" @ GOSUB 7000 @ GOTO 7280
7440 A$="*G" @ GOSUB 7000 @ GOTO 7280
7460 A$="*F" @ GOSUB 7000 @ GOTO 7280
7480 M=M+1 @ IF M>3 THEN M=1
7485 A$="*"&VAL$(M) @ GOSUB 7000 @ GOTO 7280
7500 CLEAR @ DISP "No. of Cycles and Syringe (1-3)";@
     INPUT H1,M
7510 IF M<1 OR M>3 THEN 7500
7520 A$="*"&VAL$(M) @ GOSUB 7000
7530 A$="*E"&VAL$(H1) @ GOSUB 7000
7535 IF H1 MOD 2#0 THEN C(M)=0 ELSE C(M)=W(M+5)
7540 RETURN
7700 SET TIMEOUT 7;2000 @ A$=""
7710 ON TIMEOUT 7 GOTO 7790
7720 ENTER 708 USING "#,B" ; H1@ A$=A$&CHR$(H1) @ IF
     H1=13 OR LEN(A$)>40 THEN 7740
7730 GOTO 7720
7740 OFF TIMEOUT 7 @ ON ERROR GOTO 7790
7750 C(1)=VAL(A$[10,16])
7760 C(2)=VAL(A$[20,26]) @ C(3)=VAL(A$[30,36]) @
     C(15)=1
7790 ON ERROR GOTO 1600 @ RETURN
7900 IF K=0 THEN N3=0 @ GOTO 7970
7910 ON T(19) GOTO 7920,7930,7950,7940,7960
7920 N3=4*L3^(K-1) @ GOTO 7970
7930 N3=K*300/T(5) @ IF K>T(5)/2 THEN N3=150+INT(K-
     T(5)/2)*600/T(5) @ GOTO 7970 ELSE GOTO 7970
7940 K1=T(5)+1-K @ N3=1/K1*600 @ GOTO 7970
7950 N3=5+K*100/T(5)+3*L3^(K-1) @ GOTO 7970
```

```
7960 N3=N3/L(20)
7970 W1(3)=N3*L(20)/10000
7975 W1(2)=B(7)*B(3)/T(3)*(1-B(17)*(T(5)-K)/T(5))
7980 W1(1)=B(7)-W1(2)-W1(3) @ IF I=9 THEN RETURN
7985 A$="*K"&VAL$(W1(1))&","&VAL$(W1(2))&","&VAL$
     (W1(3)) @ GOSUB 7100
7990 GOSUB 7000 @ RETURN
8000 S1(1)=4 @ S1(2)=.3 @ S1(3)=.25 @ I=9 @ A$="*WFCC8"
     @ GOSUB 7000 @ GOSUB 7700
8010 FOR K=0 TO T(5) @ GOSUB 7900 @ FOR M=1 TO 3 @
     S1(M)=S1(M)+W1(M)*T(8)*1.1 @ NEXT M @ NEXT K
8060 FOR M=1 TO 3 @ IF S1(M)>C(M) THEN DISP "Syringe
     ";M;" is filled" @ GOSUB 8100
8070 NEXT M @ RETURN
8100 K=B(20) @ A$="*"&VAL$(M) @ GOSUB 7000 @ IF M#K
     THEN 8120
8105 S1(K)=S1(K)+.2 @ IF S1(K)>W(K) THEN S1(K)=W(K)
8107 IF S1(K)>B(19) THEN S1(K)=B(19)
8110 A$="*="&VAL$(S1(K)) @ GOSUB 7000 @ W(K+5)=S1(K)
8120 A$="*G" @ GOSUB 7000
8190 RETURN
8200 I=0 ! BER. [E] [S]
8205 GOSUB 7900
8210 H2=T(3)*W1(2)/B(7)
8220 H4=T(4)*W1(3)/B(7) ! [S]
8230 H3=T1(K)*1000000/T(17)/H2 @ J1=T2(K)*1000000/
     T(18)/H2
8240 RETURN
8250 I=9 @ GOSUB 8205
8260 RETURN
8500 RESET 7
8510 ASSIGN# 1 TO A$ @ READ# 1 ; H3,H4,X0,X1,X2,X3,Y0,
     Y1,Y2,Y3,E3$,E4$
```

```
8550 GRAPH @ GCLEAR @ SCALE 0,100,0,100 @ IF X2=0 THEN
     X2=(X1-X0)/5
8560 MOVE 30,0 @ LDIR 0 @ CSIZE 3 @ LABEL E3$ @ F5=(X1-
     X0)/7 @ F6=(Y1-Y0)/10
8590 SCALE X0-F5,X1,Y0-F6,Y1 @ XAXIS Y1,X3,X0,X0+X3 @
     XAXIS Y1,X2,X0+X3,X1
8620 XAXIS Y0,X3,X0,X0+X3 @ XAXIS Y0,X2,X0+X3,X1 @
     YAXIS X1,Y3,Y0,Y0+Y3
8650 YAXIS X1,Y2,Y0+Y3,Y1 @ YAXIS X0,Y3,Y0,Y0+Y3 @
     YAXIS X0,Y2,Y0+Y3,Y1
8660 FOR K=X0+X3 TO X1 STEP X2
8670 MOVE K-F5/5,Y0-(Y1-Y0)/20
8680 LABEL VAL$(K) @ NEXT K
8690 FOR K=Y0+Y3 TO Y1 STEP Y2
8700 MOVE X0-(X1-X0)/10,K-F6/2
8710 LABEL VAL$(K) @ NEXT K
8900 ASSIGN# 1 TO * @ RETURN
9500 PRINT @ PRINT @ PRINT
9510 PRINT "Measurement ";D4$[15,20];"  ";D1$;D2 @
     PRINT "Diskette:";B9$;"  ";D3$
9540 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9542 IF H1=0 THEN 9700
9545 PRINT "Time    [S]min";TAB(16);B(13);"bis";
     TAB(24);B(14);"sec"
9546 PRINT "Time    [S]max";TAB(16);T(6);"bis";
     TAB(24);T(7);"sec"
9547 PRINT "Interval  ";L1;" / Integration";L2
9548 PRINT "Max.Stdev";B(1);" / Nonlin.";B(2);"%"
9550 PRINT "Range 1 from   ";T(1);" to";T(2);"nm"
9555 PRINT "Range 2 from   ";T(12);" to";T(13);"nm"
9557 PRINT "Int. Ref. from ";T(10);" to";T(11);"nm"
9560 PRINT "[";E1$;"]";TAB(23);T(3);"nM"
9565 PRINT "in the Assay :  ";B(17);TAB(23);B(3);"nM"
9580 PRINT "[";E2$;"]";TAB(23);T(4);"M"
```

```
9585 PRINT "Max. Conc. in Assay:  ";T(14);"M"
9600 PRINT "No. of Points";T(5);TAB(24);"Rep.";T(8)
9650 PRINT "Delta Epsilon 1 :  ";T(17);"1/mM cm" @
     PRINT "Delta Epsilon 2 :  ";T(18);"1/mM cm"
9657 PRINT "Assay Volume :";TAB(25);B(7);"ml"
9690 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9700 PRINTER IS 2 @ RETURN
```

APPENDIX I

Measure Overlay Kin 1

```
50 ! KIN 1 B.M.
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),Z8,A(5,5),B0
125 COM INTEGER C1,C2,C3,C4
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
     D3$[20],D4$[20],B9$[8]
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],F5$[1],
     E5$[60]
190 SHORT Y0,Y1,Y2,Y3,X0,X1,X2,X3,L1,L2,Q1,Q2
200 U2=60 @ F6$="0" @ N=10 @ Q1=1 @ Q2=1 @ F7$="0" @
     CLEAR @   LOCAL 7
210 F$="A" @ F5$="0" @ J=1 @ K=1 @ Z5=0 @ V1=0 @ V2=0 @
     V3=0
250 ON ERROR GOTO 320 @ PLOTTER 1 @ IF Z8=1 THEN GOTO
500
320 GOTO 800
490 ASSIGN# 1 TO * @ ASSIGN# 2 TO *
500 L1=B(11) @ L2=B(12) @ T(7)=B(16) @ T(8)=B(15) @
     L3=256^(1/T(5)) @ ON ERROR GOTO 500
505 CLEAR @ DISP "*** Messprogramm : Hauptmenu ***"
510 DISP "Dauer";B(13);B(14);"und";T(6);T(7);"sec"
520 DISP "Bereich 1 von  ";T(1);"bis";T(2);"nm"
540 DISP "Normierung von ";T(10);"bis";T(11);"nm" @
     DISP
570 DISP E1$;TAB(19);T(3);"nm (E)" @ DISP
     E2$;TAB(19);T(4);   "m (S)"
575 DISP "Stdev Mi";B(1);"% EzMsg";B(2);"%"
580 DISP "Anz Rep.";T(8);" / Anz. Punkte";T(5)
640 IF Q1=1 THEN DISP "Mit Enzym" ELSE DISP "Ohne
     Enzym"
```

```
650 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"MessM" GOTO
    1000 @ ON KEY# 3,"Motor" GOTO 860
655 ON KEY# 4,"WaschSP" GOTO 7500 @ ON KEY# 5,"MAINPG"
    GOTO 800 @ ON KEY# 6,"OhneE" GOTO 760
660 ON KEY# 7,"[E,S]" GOTO 880 @ ON KEY# 8,"Punkte"
    GOTO 940 @ KEY LABEL @ GOTO 650
760 IF Q1=1 THEN Q1=2 ELSE Q1=1
765 GOTO 500
800 CHAIN "Autost.KSYS"
820 CLEAR @ DISP "Geben Sie die ";E2$ @ DISP "Konz in M
    bei [S]<0: Referenz";
830 INPUT L(18)@ IF L(18)>T(4)/3 THEN DISP "[S] zu
    hoch" @ BEEP @ WAIT 2000 @ GOTO 820
840 MODE 0,1 @ J=0 @ GOSUB 4800 @ GOTO 1000
860 GOSUB 7250 @ GOTO 500
880 CLEAR @ DISP "Geben Sie die" @ DISP E1$;" Konz. " @
    DISP "im Assay in (nM)" @ INPUT T(3)
885 CLEAR @ DISP "Geben Sie die" @ DISP E2$;" Konz." @
    DISP "in der Spritze (M)";@ INPUT T(4)
890 GOTO 500
940 CLEAR @ DISP "Geben Sie die Anzahl Messpunkte (2-
    40) und die Anzahl Repetitionen (2-10)";
950 INPUT T(5),T(8)@ IF T(5)>40 OR T(5)<2 OR T(8)>10 OR
    T(8)<0 THEN GOTO 940 ELSE GOTO 500
1000 CLEAR @ DISP @ DISP T1$[1,63] @ ON ERROR GOTO 500
1010 DISP "Spr 1 max";B(17);"ml, Fuellst:";INT((1-
    C1/C3)*B(17)*10)/10;"ml"
1020 H1=INT((1-C2/C4)*B(19)*100)/100 @ DISP "Spr 3
    max";B(19);"ml Fuellst";H1;"ml"
1030 IF D4$[3,3]="1" THEN DISP "Spr 2 max";B(18);"ml
    Fuelg";H1/B(17)*B(18) ELSE DISP "Spr 2 off"
1050 DISP "Maximale Substratkonz";T(14);"uM"
1060 DISP "Plot: ";@ IF D4$[5,5]="1" THEN DISP "OD vs t
    Berl ";
```

```
1065 IF D4$[5,5]="2" THEN DISP "Ableit. Berl ";
1080 IF D4$[7,7]="1" THEN DISP "TN vs [Substr]";
1085 IF D4$[5,7]="000" THEN DISP "OFF" ELSE DISP " "
1090 IF D4$[4,4]="L" THEN DISP D4$[10,15];" abgezogen"
     ELSE DISP "Keine Nullinie"
1120 IF Q2=1 THEN DISP "[";E2$;"] Aufsteigend" ELSE
     DISP "[";E2$;"] Absteigend"
1150 K=1 @ GOSUB 8200 @ DISP "Assay Volumen :
     ";INT(1000*H1);"ul"
1160 IF T(19)=1 THEN DISP "Geom";
1165 IF T(19)=2 THEN DISP "Arithm";
1170 IF T(19)=3 THEN DISP "Gemischte";
1175 IF T(19)=4 THEN DISP "Reziproke";
1180 DISP " Reihe  " @ ON ERROR GOTO 1000
1200 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"Plot" GOTO
     1320 @ ON KEY# 3,"Inhalt" GOTO 1600
1230 ON KEY# 4,"Comment" GOTO 1690 @ ON KEY# 5,"Exec"
     GOTO 2000 @ ON KEY# 6,"Null" GOTO 1500
1260 ON KEY# 7,"AufAb" GOTO 1400 @ ON KEY# 8,"EinzelME"
     GOTO 820 @ KEY LABEL @ GOTO 1200
1320 IF D4$[5,7]="000" THEN D4$[5,7]="100" @ GOTO 1000
1330 IF D4$[5,7]="100" THEN D4$[5,7]="200" @ GOTO 1000
1360 IF D4$[5,7]="200" THEN D4$[5,7]="001" @ GOTO 1000
     ELSE D4$[5,7]="000" @ GOTO 1000
1400 IF Q2=1 THEN Q2=2 ELSE Q2=1
1405 GOTO 1000
1500 IF D4$[4,4]="L" THEN D4$[4,4]="0" @ GOTO 1000
1510 CLEAR @ DISP "Geben Sie die Bez.des Nulllinien
     files (3  Zeichen)";@ INPUT H1$@ D4$[4,4]="L"
1515 IF LEN(H1$)#3 THEN GOTO 1510 ELSE
     D4$[10,15]="Nul"&H1$
1540 CREATE D4$[10,15]&".KSYS",T(5)+2,768 @ ASSIGN# 1
     TO D4$[10,15]&".KSYS"
1550 FOR J=1 TO 20 @ S1(J)=T(J) @ NEXT J
```

```
1560 PRINT# 1,1 ; S1()
1570 FOR J=1 TO T(5) @ READ# 2,J ; S1()@ H1=0 @ FOR
     J1=1 TO 5 @ H1=H1+S1(J1) @ NEXT J1
1575 H1=H1/5 @ FOR J1=1 TO T(7)*2/L2+3 @ S1(J1)=S1(J1)-
     H1 @ NEXT J1 @ PRINT# 1,J+1 ; S1()
1580 NEXT J @ ASSIGN# 1 TO * @ H1$=D4$[10,15]
1590 PRINTER IS 2 @ PRINT "Nullinie als File: ";H1$;"
     gespeichert" @ RETURN
1600 CAT "."&B9$ @ DISP "Loesche File";
1605 INPUT H1$@ IF H1$="N" OR H1$="E" OR H1$=" " THEN
     GOTO 1000
1610 IF H1$="PACK" THEN PACK @ GOTO 1600 ELSE PURGE H1$
     @ GOTO 1605
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP
     "Berichtign Sie den Fehler +CONT";ERRL
1670 PAUSE
1675 GOTO 500
1690 GOSUB 1700 @ GOTO 1000
1700 CLEAR @ DISP "Kommentar 2 Zeilen";@ FLIP @ INPUT
     T1$@ FLIP
1710 T1$=T1$&" " @ IF LEN(T1$)<63 THEN GOTO 1710
1720 RETURN
1800 ASSIGN# 1 TO D4$[10,15]&".KSYS" @ READ# 1,1 ; S1()
1820 FOR J=1 TO 20 @ Z8=S1(J) @ IF J=3 OR J=4 OR J=6 OR
     J=7 OR J=15 OR J=16 THEN GOTO 1840
1835 IF Z8#T(J) THEN BEEP @ GOTO 1885
1840 NEXT J @ Z8=1 @ ASSIGN# 1 TO *
1875 H1$=D4$[10,15] @ PRINTER IS 2 @ PRINT "Nullinie:
     ";H1$;" abgezogen" @ RETURN
1885 DISP "Variable";J;T(J);"# Nullinie:";Z8
1890 DISP "Aendern Sie die Variable (999=ignore)";@
     INPUT H1@ IF H1=999 THEN 1840
1892 IF J=1 OR J=2 OR J=10 OR J=11 OR J=12 OR J=13 OR
     J=19 THEN BEEP @ GOTO 1000 ELSE T(J)=H1
```

```
1895 GOTO 1840
1900 M1=1 @ N=B(7) @ GOSUB 7000 @ WAIT 1000 @ GOSUB
     7000
1910 MODE 0,1
1920 REFERENCE 10
1930 IF NMEAS=0 THEN GOTO 1930
1940 MEASURE .1
1950 IF NMEAS=0 THEN GOTO 1950
1960 FOR J=1 TO 10 @ IF VALUE(L(J))>5 THEN GOTO 1900
1970 NEXT J @ RETURN
2000 ERASE STATUS
2010 IF T(3)=0 THEN BEEP @ DISP "[E]=0!" @ WAIT 3000 @
     GOTO 880
2020 IF T(14)<T(4)/15 OR T(14)>T(4)/4 THEN BEEP @ DISP
     "[S]      falsch!" @ WAIT 3000 @ GOTO 500
2040 GOSUB 3600 @ CLEAR @ DISP "Messung in Process "
2200 GOSUB 3650
2230 GOSUB 8000 @ F7$="0" ! TP
2239 IF Q2=2 THEN K=T(5) ELSE K=1
2240 GOSUB 7900 @ GOSUB 7800 @ IF Q2=2 THEN GOTO 2247
2245 FOR K=1 TO T(5) @ Z5=0 @ GOTO 2250
2247 FOR K=T(5) TO 1 STEP -1 @ Z5=0
2250 IF F7$="2" THEN F7$="0" @ K=P3
2260 R9=0 @ R8=0 @ S6=0 @ S7=0
2300 IF K<4 AND B(15)<5 THEN T(8)=B(15)*2 ELSE
     T(8)=B(15)
2305 FOR J=1 TO T(8) @ F6$="0"
2310 GRAPH
2350 GOSUB 3500
2360 IF F5$="2" THEN GOTO 2990
2420 GOSUB 4400
2430 GOSUB 3300 @ IF F5$="N" THEN GOTO 2500
2450 IF F5$="1" THEN GOTO 2310
2490 IF F6$="1" THEN GOTO 2530
```

```
2500 NEXT J
2530 GOSUB 3700
2580 IF F6$="2" THEN GOTO 2905
2590 GOSUB 3800
2654 IF F7$="1" THEN GOTO 2990
2655 NEXT K @ GOTO 2990 ! ENDM
2900 IF F$="A" THEN GOTO 2250
2905 ON KEY# 2,"N.Stdev" GOTO 2950 @ ON KEY# 3,"EinzW"
     GOTO 2960 @ ON KEY# 4,"AlleW" GOTO 2250
2910 ON KEY# 6,"Weiter" GOTO 2590 @ CLEAR @ KEY LABEL
2920 BEEP 300,10 @ WAIT 3000 @ GOTO 2920
2950 F6$="0" @ DISP "Neue Standardabw. 2 Eing." @ INPUT
     B(1),B(2)@ GOTO 2530
2960 F6$="1" @ DISP "Einzelmessung NR." @ INPUT J@ GOTO
     2310
2990 ON ERROR GOTO 3000 @ T(7)=B(16) @ L1=B(11) @
     L2=B(12) @ T(8)=B(15) @ ASSIGN# 1 TO *
2995 DISP T1$ @ DISP @ DISP
3000 ON KEY# 3,"Punkt W" GOTO 3100 @ ON KEY#
     2,"Ausgabe" GOTO 3250 @ ON ERROR GOTO 3000
3010 ON KEY# 8,"Motor" GOTO 3050 @ ON KEY# 4,"Comment"
     GOTO 3030 @ ON KEY# 1,"Exit" GOTO 490
3015 ON KEY# 5,"ResM" GOTO 3060 @ ON KEY# 6,"*Schluss"
     GOTO 3000 @ ON KEY# 7,"-Menu*" GOTO 3000
3017 KEY LABEL
3020 BEEP 300,10 @ WAIT 3000 @ GOTO 3020
3030 GOSUB 1700 @ GOTO 2995
3050 GOSUB 7250 @ GOTO 3000
3060 F7$="2" @ CLEAR @ DISP "Ab welchem Punkt
     weiterfahren" @ INPUT K@ P3=K @ GOSUB 3110
3070 GOTO 2240
3100 F7$="1" @ CLEAR @ DISP "Welchen Punkt wiederholen"
     @ INPUT K@ GOSUB 3110 @ GOTO 2250
3110 T1(K)=0 @ T2(K)=0
```

```
3150 ASSIGN# 1 TO "TEMP1.KSYS"
3200 RETURN
3250 IF Q1=2 THEN GOSUB 1510 @ GOTO 4200 ELSE GOTO 4200
3300 S7=0 @ F5$="0" ! EM TEST
3310 FOR J1=T8/L2+1 TO T9/L2+1 @ H1=(S3(J1+1)+S3(J1-
     1))/2-S3(J1) @ IF ABS(H1)>S7 THEN S7=H1
3320 NEXT J1
3340 IF S7>.002*(1+2*K/T(5))*B(2) THEN DISP
     "Msg";K;J;"wiederh." @ F5$="1" ELSE GOTO 3350
3345 IF Z5>3*T(8) THEN PRINT "Msg ";K;J;" ungenau" @
     Z5=Z5-2 @ F5$="N" ELSE Z5=Z5+1 @ BEEP
3350 IF S6=0 THEN S6=1
3360 RETURN
3400 S7=0 @ F5$="0" ! MI TEST
3410 I3=0 @ FOR J1=T8/L2+1 TO T9/L2+1 @ H1=(S9-
     S1(J1))^2 @ I3=I3+1 @ S7=S7+H1
3420 NEXT J1 @ S7=S7/I3 @ S6=S7^.5 @ IF S6<.05*(1.5-
     K/T(5))*B(1) OR Q1=2 THEN GOTO 3450
3440 IF S6>ABS(S9/25*B(1)) THEN DISP "Mittel";K;" nicht
     1.Ordnung" @ BEEP ELSE GOTO 3450
3445 IF Z5>3*T(8) THEN PRINT "Wert unzuverlaessig" @
     F5$="0" @ Z5=Z5-1 ELSE Z5=Z5+1 ! @ F5$="1"
3450 IF S6=0 THEN S6=1
3460 RETURN
3480 IF ERRN=2 THEN GOSUB 1900 ELSE GOTO 1660
3490 PRINT "Ref W." @ GOTO 2310
3500 GOSUB 7900 ! MESSE
3510 H4=(T(11)-T(10))/2+1 @ H2=(T(2)-T(1))/2+1 @ ON
     ERROR GOTO 3480
3515 L1=INT((B(11)*10-3)*K/T(5)+4)/10 @
     L2=INT((B(12)*10-3)*K/T(5)+4)/10
3516 IF L2>B(12) THEN L2=B(12)
3520 T(7)=B(16)/B(12)*L2 @ IF L1>B(11) THEN L1=B(11)
3530 MODE 0,1 @ GOSUB 7800
```

```
3540 MEASURE L1,L2,0,T(7)*2+2*L2
3550 FOR J1=1 TO T(7)*2/L2+3
3560 IF NMEAS<J1 THEN GOTO 3560
3570 H1=0 @ FOR I2=T(1) TO T(2) STEP 2 @
     H1=H1+VALUE(I2) @ NEXT I2
3580 S2(J1)=H1/H2 @ H1=0
3585 FOR I2=T(10) TO T(11) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
3590 S2(J1)=S2(J1)-H1/H4
3595 NEXT J1 @ RETURN
3600 ON ERROR GOTO 3600 ! Setup
3601 CLEAR @ DISP "Bezeichnung der Messung zB A,1";@
     INPUT D1$,D2@ IF D2>9 OR D2<0 THEN 3601
3602 DISP "Memocode (4 Buchstaben)";@ INPUT D4$[17,20]
3603 D$=D4$[17,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
3605
3604 ASSIGN# 1 TO D$ @ DISP "File existiert" @ ASSIGN#
     1 TO * @ BEEP @ WAIT 2000 @ GOTO 1000
3605 IF ERRN=130 OR ERRN=125 THEN DISP B9$;"nicht gef"
     @ BEEP @ WAIT 2000 @ GOTO 3600
3608 ON ERROR GOTO 1660
3610 IF D4$[5,7]#"000" THEN DISP "Schalten Sie den
     Plotter ein"
3615 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
3620 PLOTTER 1 @ PRINTER 2
3625 FOR I=1 TO 15 @ K=T(5) @ GOSUB 8200 @
     L(20)=L(20)*T(14)/H4 @ NEXT I
3627 L(18)=-1 @ GOSUB 4800 ! RE
3630 L(18)=T(14)/3 @ GOSUB 4800
3635 GOSUB 8000 @ GOSUB 9500
3640 FOR K=0 TO 40 @ T1(K)=0 @ T2(K)=0 @ NEXT K
3645 IF D4$[4,4]="L" THEN GOSUB 1800
3647 RETURN
```

```
3650 PRINT @ PRINT "NR   Konst. 1.0   Rel.Abw" @
     F6$="0"
3655 DISP "Keys nur zwischen Toenen aktiv" @
     H1=INT(T(8)*T(5)*(T(7)+5)/200+1)*12
3657 DISP "Messdauer";H1;"Min."
3660 DISP "NR   Konst 1.0" @ DISP "NR  [Substr]
     KorrFa"
3670 ASSIGN# 2 TO "TEMP.KSYS" @ ASSIGN# 1 TO
     "TEMP1.KSYS"
3680 ON KEY# 1,"Exit" GOTO 490 @ RETURN
3700 FOR J1=1 TO 80 @ S2(J1)=0 @ NEXT J1 ! MITTEL
3710 FOR J=1 TO T(8) @ READ# 1,J ; S1()
3720 FOR J1=1 TO T(7)*2/L2+3 @ S2(J1)=S2(J1)+S1(J1) @
     NEXT J1
3730 NEXT J @ FOR J=1 TO 80 @ S1(J)=0 @ NEXT J @ H5=H4
3735 IF D4$[4,4]="L" THEN ASSIGN# 1 TO * @ ASSIGN# 1 TO
     D4$[10,15]&".KSYS" @ READ# 1,K+1 ; S1()
3740 FOR J1=1 TO T(7)*2/L2+3 @ S2(J1)=S2(J1)/T(8)-
     S1(J1) @ NEXT J1 @ GOSUB 4400
3750 PRINT# 2,K ; S2() @ IF D4$[4,4]="L" THEN ASSIGN# 1
     TO * @ ASSIGN# 1 TO "TEMP1.KSYS"
3760 GOSUB 3400 @ T1(K)=S9 @ T2(K)=S6 @ GOSUB 8200 @
     H2=H1/(H1-N1*B(17)/C3)
3780 DISP USING "DD,XX,4D.DD,3X,D.4D,3X,D.4D" ;
     K,H4,H2,S6 @ RETURN
3795 ! AUSG
3800 X6=0 @ F5=1 @ F6=1 @ X5=1 @ IF D4$[5,5]="2" THEN
     X6=1
3805 IF Q1=2 THEN F6=2
3810 H3=1 @ GOSUB 6500
3820 X5=0 @ IF D4$[5,5]="1" THEN F5=J @ F6=2 @ X6=1 @
     H3=1 @ GOSUB 6500
3830 IF ABS(T1(K))>9 OR ABS(T2(K))>9 THEN PRINT
     K;T1(K);T2(K) @ GOTO 3850
```

```
3840 PRINTER IS 2 @ PRINT USING "DD,3X,SD.6D,3X,SD.6D"
     ; K,T1(K),T2(K)
3850 RETURN
3990 ! AUSGABE
4200 PRINTER 2 @ PRINT @ IF Q1=1 THEN PRINT
     TAB(7);"d(ln(ODt+dt-ODt)/dt"
4201 IF Q1=2 THEN PRINT TAB(7);"Absorption";
     T(1);"bis";T(2);"nm"
4202 GRAPH @ COPY @ PRINT @ PRINT @ PRINT @ PRINT @
     PRINT
4203 PRINT "Messung  ";D1$;"  ";D2;"     ";D4$[17,20]
4205 PRINT @ PRINT @ PRINT "NR.    Ber 1   Abs.Abw
     [Substr]"
4210 PRINT "    TN(1/sec) TN(1/sec)  M" @ PRINT
4220 K=T(5) @ GOSUB 8200
4230 X5=1 @ X6=0 @ E5$="PFTN.KSYS" @ IF D4$[7,7]="1"
     THEN X6=1
4250 GOSUB 8500
4260 LORG 0 @ CSIZE 3 @ GOSUB 5500 @ ON ERROR GOTO 1660
4270 FOR K=0 TO T(5) @ GOSUB 8200
4280 H2=T(3)*(H1-N3*B(19)/C3)/H1 @ H3=-
     (T1(K)*1000*H4/H2) @ J1=ABS(T2(K)*1000*H4/H2)
4290 MOVE H4,H3 @ LABEL "*"
4300 PRINTER IS 2 @ IF ABS(H3)>999 OR ABS(J1)>999 THEN
     PRINT K;H3,J1,H4 @ GOTO 4312
4310 PRINT USING "DD,XXX,SDDD.D,XXX,SDDD.D,XXX,DDDD.DD"
     ; K,H3,J1,H4
4312 PRINT# 1 ; H3,J1,H4
4315 IF D4$[7,7]="1" THEN GOSUB 6700
4320 NEXT K @ PRINTER IS 2 @ PRINT @ PRINT @ PRINT
     TAB(7);E4$ @ COPY
4330 PRINT @ PRINT T1$ @ PRINT @ PRINT @ PRINT @ PRINT
     @ PRINT
4335 IF T(9)#3 THEN GOTO 4350
```

```
4340 FOR K=1 TO T(5) @ READ# 2,K ; S1()
4345 FOR J=1 TO T(7)*2/L2+2 @ PRINT# 1 ; S1(J) @ NEXT J
     @ NEXT K
4350 ASSIGN# 2 TO * @ ASSIGN# 1 TO * @ GOTO 1000
4400 T8=INT((B(13)+K/T(5)*(T(6)-B(13)))/L2)*L2 @ ALPHA
     @ ON ERROR GOTO 1660
4402 T9=INT((B(14)+K/T(5)*(B(16)-B(14)))/L2)*L2
4405 BEEP 100,10 @ ON KEY# 2,"N.Stdev" GOTO 4560 @ ON
     KEY# 5,"End Men" GOTO 2990
4410 ON KEY# 4,"Alle W" GOTO 2250 @ IF J=T(8) THEN ON
     KEY# 3,"EinzW" GOTO 4570
4450 FOR J1=1 TO T(7)/L2+2 @ J2=INT(J1+T(7)/L2)
4455 H3=ABS((S2(J2+1)+S2(J2)+S2(J2-1))/3-S2(J1)) @ IF
     H3<.0001 THEN H3=1
4460 S1(J1)=LOG(H3) @ S3(J1)=S2(J1)-S2(J2)
4465 NEXT J1
4470 S9=0 @ S8=0 @ I1=0 @ H1=S1(1) @ H2=S2(1)
4475 FOR J1=2 TO T(7)/L2+1 @ H3=S1(J1) @ H4=S3(J1)
4480 S1(J1)=(S1(J1+1)-H1)/(2*L2) @ S3(J1)=(S3(J1+1)-
     H2)/(2*L2) @ IF J1>T9/L2+1 THEN GOTO 4490
4485 IF J1>=T8/L2+1 THEN I1=I1+1 @ S9=S9+S1(J1)
4490 H1=H3 @ NEXT J1 @ I3=0 @ S9=S9/I1 @ IF S9=0 THEN
     S9=1
4500 BEEP 200,10 @ OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4
     @ OFF KEY# 5 @ GOTO 4600
4560 DISP "Neue Standardabw 2 Eing.";@ INPUT B(1),B(2)@
     GOTO 4400
4570 F6$="2" @ GOTO 4400
4600 IF J>10 THEN GOTO 4650
4630 DISP USING "DD,X,DD,XX,SDD.4D,XXX" ; K,J,S9 @ IF
     J=0 THEN GOTO 4650
4640 PRINT# 1,J ; S2()
4650 RETURN
4800 F6$="0" ! TEST m
```

```
4810 LAMBDA L(1),L(2),L(3),L(4),L(5),L(6),L(7),L(8),
     L(9),L(10)
4820 TIME SCALE 0 TO T(7)
4830 ABSORBANCE
4840 IF L(18)<=0 THEN GOTO 1900
4850 N3=100 @ Z8=T(19) @ T(19)=5 @ K=T(5)
4860 FOR I=1 TO 10 @ GOSUB 8200 @ N3=N3*L(18)/H4 @ NEXT
     I @ T(19)=Z8 @ Z8=1 @ GOSUB 7800
4865 WAIT 1000 @ GOSUB 3510
4920 H1=0 @ H2=0 @ H3=0
4930 H1=S2(1) @ H2=S2(T(7)/L2+2) @ H3=S2(T(7)*2/L2+2) @
     H4=ABS((H1-H2)/(H1-H3)*100)
5010 GOSUB 8500 @ DISP USING "13A,SD.5D" ; "Anfangs OD
     : ",H1 @ H1=H1/T(15)*1000
5020 DISP USING "12A,3X,S3D.3D,X,2A" ;
     "[S]Gemessen:",H1,"uM"
5030 DISP USING "18A,3X,3D.D,A" ; "Reaktion im
     1.Teil",H4,"%"
5050 DISP "Kinetische Konst. 1.Ordnung"
5060 X5=1 @ X6=0 @ F5=1 @ F6=2 @ IF D4$[5,5]="1" THEN
     X6=1
5070 H3=1 @ GOSUB 6500
5080 FOR J1=1 TO T(7)/L2+1 @ J2=J1+T(7)/L2+1 @
     S1(J1)=S2(J2) @ NEXT J1
5100 X5=1 @ X6=0 @ F5=1 @ F6=1 @ IF D4$[5,5]="1" THEN
     X6=1
5110 H3=1 @ GOSUB 6500
5150 K=INT(T(5)/3) @ J=0 @ GOSUB 4400 @ GOSUB 3400 @
     H4=ABS(S6/S9*100) @ IF H4>9999 THEN H4=9999
5160 DISP USING "20A,3X,4D.D,A" ; "Genauigkeit
     (Noise):",H4,"%" @ H1=0 @ H2=0
5170 FOR J=2 TO 6 @ H1=H1+S1(J) @ NEXT J @ H1=H1/5
5180 FOR J=T(7)/L2-3 TO T(7)/L2+1 @ H2=H2+S1(J) @ NEXT
     J @ H2=H2/5 @ H3=ABS((H1-H2)/H1*100)
```

```
5190 DISP USING "21A,2X,4D.D,A" ; "Unlinearitaet von
     k1:";H3;"%"
5300 DISP @ DISP @ GOSUB 5400 @ IF Q1=2 THEN RETURN
5350 H1=P1 @ H2=P2 @ X6=0 @ X5=1 @ E5$="PFAB.KSYS" @ IF
     D4$[5,5]="2" THEN X6=1
5355 GOSUB 8500 @ IF D4$[5,5]="1" OR D4$[6,6]="1" THEN
     P1=H1 @ P2=H2
5360 X5=1 @ X6=0 @ F5=1 @ F6=1 @ IF D4$[5,5]="2" THEN
     X6=1
5365 H3=1 @ GOSUB 6500
5390 GOSUB 5400 @ RETURN
5400 ALPHA @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8 @ ON
     KEY# 2,"Forts" GOTO 5490
5410 ON KEY# 3,"Graph" GOTO 5420 @ ON KEY# 4,"Alpha"
     GOTO 5400 @ KEY LABEL @ GOTO 5450
5420 GRAPH
5450 GOTO 5450
5490 RETURN
5500 D$=D4$[17,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     1660
5510 IF T(9)=3 THEN H1=INT(T(5)*T(7)/L2/16)+10 ELSE
     H1=10
5550 CREATE D$,H1,256 @ ASSIGN# 1 TO D$
5570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B()
     @ D2=10
5660 RETURN
6500 IF X6=0 AND X5=0 THEN GOTO 6690
6510 IF X6=1 THEN PRINTER IS 705 @ PRINT "PU;" @ GOSUB
6900
6515 IF X5=1 THEN PLOT 0,0,2
6520 FOR J1=2 TO T(7)/L2+1
6530 H4=(J1-1)*L2
6540 ON F6 GOTO 6550,6560,6570
6550 H3=S1(J1) @ GOTO 6580
```

```
6560 H3=S2(J1) @ GOTO 6580
6570 H3=S3(J1)
6580 IF X6=1 THEN PRINT "PA";INT(H4*P1);",
     ";INT(H3*P2);";PD;"
6585 IF X5=1 THEN PLOT H4,H3,1
6590 NEXT J1 @ IF X6=1 THEN PRINT "PU;"
6595 IF X5=1 THEN PENUP @ PLOT 0,0,2
6690 RETURN
6700 PRINTER IS 705 @ PRINT "SP 1;"
6705 H1=INT(J1*P2) @ H3=INT(H3*P2) @ H4=INT(H4*P1)
6710 H2$="*"&CHR$(3) @ PRINT "PU;PA";H4;",";H3;";
     LB";H2$;";"
6720 H2$="+"&CHR$(3) @ PRINT "PU;PA";H4;",";H1;";
     LB";H2$;";"
6730 IF K=T(5) THEN PRINT "PU;PA0,0;"
6740 RETURN
6900 IF F5=1 THEN PRINT "LT;"
6910 IF F5=2 THEN PRINT "LT1,1;"
6920 IF F5=3 THEN PRINT "LT2,1;"
6930 IF F5=4 THEN PRINT "LT3,2;"
6940 IF F5=5 THEN PRINT "LT4,4;"
6950 PRINT "SP";H3;";" @ RETURN
7000 IF ABS(N)>30000 THEN RETURN
7010 IF N=0 THEN RETURN
7020 IF M1=1 THEN C1=C1+N @ M3=INT(15000/B(5)) ELSE
     C2=C2+N @ M3=INT(15000/B(6))
7021 GOSUB 7100 @ IF C1>C3+90 THEN C1=C1-N @ DISP
     "Motor 1 out of Range" @ GOTO 7090
7022 IF C2>C4+90 THEN C2=C2-N @ DISP "Motor 2 out of
     Range" @ GOTO 7090
7023 IF C1<-90 THEN C1=C1-N @ DISP "Motor 1 out of
     Range" @ GOTO 7090
7024 IF C2<-90 THEN C2=C2-N @ DISP "Motor 2 out of
     Range" @ GOTO 7090
```

```
7025 IF N>0 THEN M2=129 ELSE M2=128
7030 N=ABS(N)
7050 OUTPUT 709 USING "#,B" ; 0
7080 SAVE M1,M2,M3,N
7090 RETURN
7100 IF V1=0 THEN M3=M3+75
7110 IF V1=0 AND D4$[3,3]="1" THEN V1=2
7115 IF V3=V1+V2 THEN GOTO 7190
7116 OUTPUT 709 USING "#,B" ; 0
7120 REMOTE 7 @ V3=V1+V2
7130 OUTPUT 7 USING "#,B" ; V3,V3+64,V3 @ WAIT 500
7180 LOCAL 7
7190 RETURN
7250 M1=1 ! MOTOTST ***
7280 ON KEY# 1,"Ausg" GOTO 7430 @ ON KEY# 2,"Eins" GOTO
     7420
7300 ON KEY# 3,"Zurk" GOTO 7400 @ ON KEY# 4,"Empty"
     GOTO 7460
7320 IF M1=1 THEN ON KEY# 5,VAL$(INT(N/C3*B(17)*100)
     /100) GOTO 7360
7325 IF M1<0 THEN ON KEY# 5,VAL$(INT(N/C4*B(19)*1000)
     /1000) GOTO 7360
7330 IF M1=1 THEN ON KEY# 6,VAL$(INT((1-
     C1/C3)*B(17)*10)/10) GOTO 7440
7335 IF M1<0 THEN ON KEY# 6,VAL$(INT((1-
     C2/C4)*B(19)*100)/100) GOTO 7440
7340 ON KEY# 7,"Retur" GOTO 7355
7350 IF M1=1 THEN ON KEY# 8,"Mot A" GOTO 7480 ELSE ON
     KEY# 8,"Mot B" GOTO 7480
7352 KEY LABEL @ GOTO 7280
7355 OFF KEY# 1 @ V1=16 @ GOSUB 7100 @ RETURN
7360 CLEAR @ DISP "Volumen pro Key";@ INPUT H3@ IF M1=1
     THEN N=INT(H3*C3/B(17)) @ GOTO 7280
7370 IF M1<0 THEN N=INT(H3*C4/B(19)) @ GOTO 7280
```

```
7400 V1=12 @ IF M1=1 THEN V1=3
7410 GOSUB 7000 @ GOTO 7280
7420 IF N>0 THEN N=-N
7425 V1=12 @ IF M1=1 THEN V1=3
7427 GOSUB 7000 @ GOTO 7280
7430 V1=0 @ GOSUB 7000 @ GOTO 7280
7440 IF M1=1 THEN V1=3 @ N=C3-B(3)-C1 ELSE V1=12 @
     N=C4-B(4)-C2
7441 IF N<-30000 THEN N=-30000 @ GOSUB 7000 @ GOTO 7440
7445 GOSUB 7000 @ GOTO 7280
7460 IF M1=1 THEN V1=3 @ N=C3-C1 ELSE V1=12 @ N=C4-C2
7461 IF N>30000 THEN N=30000 @ GOSUB 7000 @ GOTO 7460
7465 GOSUB 7000 @ GOTO 7280
7480 IF M1=1 THEN M1=-1 @ GOTO 7280 ELSE M1=1 @ GOTO
7280
7500 CLEAR @ DISP "Anzahl Zyklen und Spritze (1/2)";@
     INPUT H1,H2@ M1=1 @ IF H2=2 THEN M1=-1
7520 FOR I1=.5 TO H1 STEP .5
7530 IF M1=1 THEN V1=3 @ N=C3-C1 ELSE V1=12 @ N=C4-C2
7534 IF N>30000 THEN N=30000 @ GOSUB 7000 @ GOTO 7530
7535 GOSUB 7000
7537 IF I1=H1 THEN GOTO 500
7538 I1=I1+.5 @ IF I1=H1 THEN GOSUB 7440 @ GOTO 500
7540 IF M1=1 THEN N=(C3-B(3)-C1)/4 ELSE N=(C4-B(4)-
     C2)/4
7545 GOSUB 7000 @ NEXT I1 @ GOTO 500
7800 N1=INT(N1) @ N3=INT(N3) @ C1=C1+N1 @ C2=C2+N3 @
     U1=129 @ U2=B(20)
7842 IF C1>C3 THEN C1=C1-N1 @ C2=C2-N3 @ DISP "Motor 1
     out of Range" @ GOTO 7895
7844 IF C2>C4 THEN C1=C1-N1 @ C2=C2-N3 @ DISP "Motor 2
     out of Range" @ GOTO 7895
7850 V1=0 @ GOSUB 7100
7860 OUTPUT 709 USING "#,B" ; 0
```

```
7880 GET U1,U2,U3,U4
7890 RETURN
7895 F5$="2" @ GOTO 7890
7900 N1=B(7) @ IF K=0 THEN N3=0 @ GOTO 7970
7910 ON T(19) GOTO 7920,7930,7950,7940,7960
7920 N3=4*L3^(K-1) @ GOTO 7965
7930 N3=K*300/T(5) @ IF K>T(5)/2 THEN N3=150+INT(K-
     T(5)/2)*600/T(5) @ GOTO 7970 ELSE GOTO 7965
7940 K1=T(5)+1-K @ N3=1/K1*600 @ GOTO 7965
7950 N3=K*200/T(5)+3*L3^(K-1) @ GOTO 7965
7960 N3=N3/L(20) @ GOTO 7970
7965 H1=1-N3/(40*N1) @ N3=N3*H1 @ N1=N1*H1
7970 N3=N3*L(20) @ IF D4$[3,3]="1" THEN N1=N1*(1-
     B(18)/B(17))
7972 U4=INT(10240/N1) @ IF N3#0 THEN U3=INT(10240/N3) @
     N3=10240/U3 ELSE U3=10242 @ N3=0
7975 N1=10240/U4 @ RETURN
8000 H2=100 @ H3=B(7) ! TOT PUL
8010 FOR K=0 TO T(5) @ GOSUB 8200 @ H2=H2+T(8)*INT(N3)
     @ H3=H3+T(8)*INT(N1) @ NEXT K
8060 IF H2>C3-C1 THEN BEEP @ CLEAR @ DISP "Nicht genug
     Volumen in Spritze 1" @ GOTO 8150
8070 IF H3>C4-C2 THEN BEEP @ CLEAR @ DISP "Nicht genug
     Volumen in Spritze 2" @ GOTO 8150
8140 OFF KEY# 7 @ RETURN
8150 DISP "KEY# 1=Exit und Key# 7= Fortsetzung " @ ON
     KEY# 7,"Forts" GOTO 8140
8160 BEEP 400,30 @ WAIT 1000 @ GOTO 8160
8200 GOSUB 7900 ! CALC PULS
8210 H1=N3*B(19)/C4+N1*B(17)/C3 @ IF D4$[3,3]="1" THEN
     H1=H1+N1*B(18)/C3
8220 H4=T(4)*N3*B(19)/C4/H1
8230 RETURN
8500 ! AXES
```

```
8520 ASSIGN# 1 TO E5$ @ READ# 1 ; P1,P2,X0,X1,X2,
     X3,Y0,Y1,Y2,Y3,E3$,E4$
8540 IF X5=0 THEN GOTO 8750
8550 GRAPH @ GCLEAR @ SCALE 0,100,0,100 @ IF X2=0 THEN
     X2=(X1-X0)/5
8560 MOVE 30,0 @ LDIR 0 @ CSIZE 3 @ LABEL E3$ @ F5=(X1-
     X0)/7 @ F6=(Y1-Y0)/10
8590 SCALE X0-F5,X1,Y0-F6,Y1 @ XAXIS Y1,X3,X0,X0+X3 @
     XAXIS Y1,X2,X0+X3,X1
8620 XAXIS Y0,X3,X0,X0+X3 @ XAXIS Y0,X2,X0+X3,X1 @
     YAXIS X1,Y3,Y0,Y0+Y3
8650 YAXIS X1,Y2,Y0+Y3,Y1 @ YAXIS X0,Y3,Y0,Y0+Y3 @
     YAXIS X0,Y2,Y0+Y3,Y1
8660 FOR K=X0+X3 TO X1 STEP X2
8670 MOVE K-F5/5,Y0-(Y1-Y0)/20
8680 LABEL VAL$(K) @ NEXT K
8690 FOR K=Y0+Y3 TO Y1 STEP Y2
8700 MOVE X0-(X1-X0)/10,K-F6/2
8710 LABEL VAL$(K) @ NEXT K
8750 IF X6=0 THEN GOTO 8900
8760 ON ERROR GOTO 8900
8770 PRINTER IS 705 @ H1$=CHR$(3)
8780 FOR I1=1 TO 5 @ READ# 1 ; E5$@ PRINT E5$ @ NEXT I1
8790 PRINT "PA3000,7200;SI.25,.3;CP0,0;LB";
     T1$[1,32];H1$;";"
8795 PRINT "PA3000,6900;LB";T1$[33,63];H1$;";"
8800 PRINT "PA4500,600;LB";E3$;H1$;";"
8810 PRINT "PA600,2500;DI0,1;LB";E4$;H1$;";"
8820 READ# 1 ; E5$@ PRINT E5$ @ GOTO 8820
8830 ON ERROR GOTO 1660
8900 ASSIGN# 1 TO * @ RETURN
9500 PRINTER IS 2 @ PRINT @ PRINT @ PRINT @ PRINT
     "Messung  ";D1$;"  ";D2;"    ";D4$[17,20]
9530 PRINT "Datum  ";D3$
```

```
9540 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9545 PRINT "Messdauer 1 von";TAB(16);B(13);"bis";
     TAB(24);B(14);"sec"
9546 PRINT "Messdauer 2 von";TAB(16);T(6);"bis";
     TAB(24);T(7);"sec"
9547 PRINT "Intervall ";L1;" / Integration";L2
9548 PRINT "Max. relative Standardabw ";B(1);"%"
9550 PRINT "Bereich 1 von    ";T(1);"bis";T(2);"nm"
9557 PRINT "Normierung   von ";T(10);"bis";T(11);"nm"
9560 PRINT "[";E1$;"]";TAB(24);T(3);"nM"
9580 PRINT "[";E2$;"]";TAB(24);T(4);"M"
9600 PRINT "Anzahl der Messpunkte";TAB(29);T(5)
9610 PRINT "Anzahl Repetitionen  ";TAB(30);T(8)
9640 PRINT "Spr 1: ";B(17);"ml  /  Spr 3: ";B(19);"ml"
9645 IF D4$[3,3]="1" THEN PRINT "Spritze 2  angeschl.
     ";B(18);"ml"
9657 PRINT "Assay Volumen : ";TAB(25);INT(1000*B(17)*
     B(7)/C3);"ul"
9690 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9700 RETURN
```

APPENDIX J
Measure Overlay KINSU

```
1 ! KINSU B.MICHEL
10 ! **************************
20 ! **************************
30 ! **************************
40 ! **************************
50 ! **************************
60 ! **************************
70 ! **************************
80 ! **************************
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
     L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
     D3$[20],D4$[20],B9$[8]
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],F5$[1],
     A$[60],H1$[20]
195 INTEGER I,I2,I5,I6
200 F6$="0" @ Q1=1 @ Q2=1 @ F7$="0" @ CLEAR @ I=0
210 F$="A" @ F5$="0" @ J=1 @ K=1 @ D4$[5,5]="1"
250 ON ERROR GOTO 800 @ PLOTTER 1 @ IF C(8)#1 THEN GOTO
     800
500 L1=B(11) @ L2=B(12) @ L3=256^(1/T(5)) @ ON ERROR
     GOTO 500
510 CLEAR @ DISP "    Km Measure : Main Menu" @ DISP
520 DISP E1$;TAB(19);T(3);"nm (E)" @ DISP
     E2$;TAB(19);T(4);"m (S)" @ DISP
530 DISP "No. of Rep.";T(8) @ DISP "No. of Points
     ";T(5)
650 ON KEY# 1,"DiVar" GOTO 720 @ ON KEY# 2,"MeasM" GOTO
     1000 @ ON KEY# 3,"Syringe" GOTO 860
```

```
655 ON KEY# 4,"Wash" GOTO 700 @ ON KEY# 5,"MAINPG" GOTO
     800 @ ON KEY# 6,"TestM" GOTO 790
660 ON KEY# 7,"[E,S]" GOTO 880 @ ON KEY# 8,"Points"
     GOTO 940 @ KEY LABEL @ GOTO 650
700 GOSUB 7500 @ GOTO 500
720 PRINTER IS 1 @ GOSUB 9545 @ PAUSE
725 GOTO 500
760 CLEAR @ IF B(4)=0 THEN B(4)=1 @ DISP "Enter v in
     assay";@ INPUT B(10)@ GOTO 765
762 B(4)=0 @ DISP "Enter [E] in Assay";@ INPUT B(5)
765 B(3)=B(5) @ GOTO 1000
790 RESET 7 @ CHAIN "KONZMSG.KSYS"
800 RESET 7 @ CHAIN "Autost.KSYS"
820 CLEAR @ DISP "** Single Meas **"
825 DISP E2$ @ DISP "Conc in M at [S]<0: Reference";
830 INPUT B(6)@ C(10)=1
835 DISP E1$ @ DISP "Conc in nM (Stock=";T(3);"nM)";@
     INPUT B(5)
840 B(3)=B(5) @ MODE 0,1 @ GOSUB 4800 @ GOTO 1000
860 GOSUB 7250 @ GOTO 500
880 CLEAR @ DISP E1$;" Conc. in (nM)"
885 DISP "and ";E2$;" Conc. in (M)" @ INPUT T(3),T(4)
890 GOTO 500
900 CLEAR @ IF B(20)=2 THEN DISP E1$;ELSE DISP E2$;
905 DISP " Conc. Nr.";C(6)
910 DISP "in Syringe (C=clear Buffer)" @ DISP "negative
     [E] = decreasing [S]" @ INPUT H1$
915 IF H1$="C" THEN C(6)=1 @ GOTO 900
920 E(C(6))=VAL(H1$) @ C(6)=C(6)+1 @ RETURN
940 CLEAR @ DISP "No. of Points (2-40) and Rep. (2-
     10)";
950 INPUT T(5),T(8)@ IF T(5)>40 OR T(5)<2 OR T(8)>10 OR
     T(8)<0 THEN GOTO 940 ELSE GOTO 500
```

```
1000 CLEAR @ DISP "    Km Measure Menu" @ DISP @ DISP
     T1$[1,63] @ DISP @ ON ERROR GOTO 500
1020 DISP "Syr.1:";INT(C(1)*10)/10;"2:";INT(C(2)*
     100)/100;"3:";INT(C(3)*100)/100;"ml" @ DISP
1050 DISP "Max. [S]  :  ";T(14);"uM"
1090 IF D4$[4,4]="L" THEN DISP "Blank ";D4$[6,14];"
     subtracted" ELSE DISP "No Blank"
1120 IF B(4)=1 THEN DISP "[E] Autom."
1130 IF D4$[5,5]="1" THEN DISP "Print On" ELSE DISP
     "Print OFF"
1140 IF C(9)=0 THEN DISP "No Reference" @ BEEP
1150 DISP "Assay Volume : ";B(7);"ul" @ ON ERROR GOTO
     1000
1200 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"Print" GOTO
1280 @ ON KEY# 3,"Content" GOTO 1600
1230 ON KEY# 4,"Comment" GOTO 1690 @ ON KEY# 5,"KmMeas"
     GOTO 1300 @ ON KEY# 6,"Blank" GOTO 1500
1260 ON KEY# 7,"[E]auto" GOTO 760 @ ON KEY#
     8,"SingleMe" GOTO 820 @ KEY LABEL @ GOTO 1200
1280 IF D4$[5,5]="0" THEN D4$[5,5]="1" ELSE
     D4$[5,5]="0"
1290 GOTO 1000
1300 ON ERROR GOTO 1000 @ IF C(9)=0 THEN GOTO 1000
1302 CLEAR @ DISP "Series and Number eg A,1";@ INPUT
     D1$,B1@ IF B1>999 OR B1<0 THEN 1300
1304 DISP "Memocode (6 Characters)";@ INPUT D4$[15,20]@
     A$="*4" @ GOSUB 7000
1310 M=B(20) @ IF C(M)<W(M)/8 THEN H1=7 @ GOSUB 7510
1320 D2=B1 @ B1=B1+1 @ F8$="N"
1325 IF C(6)=1 THEN GOSUB 900 ELSE H1=0 @ PRINTER IS 1
     @ GOSUB 9500
1330 B(3)=B(5) @ IF E(1)#0 THEN Q1=1 @ GOTO 1335
1332 E(1)=1 @ B(3)=.1 @ Q1=2 @ D4$[4,4]="0" @
     D1$=CHR$(NUM(D1$)+1) @ D2=0
```

```
1335 IF E(1)<0 THEN Q2=2 @ E(1)=-E(1) ELSE Q2=1
1337 IF B(20)=2 THEN T(3)=E(1) ELSE T(4)=E(1)
1340 FOR I=1 TO C(6)-1 @ E(I)=E(I+1) @ NEXT I @
     C(6)=C(6)-1
1350 W1(2)=B(7)*B(5)/T(3) @ GOSUB 8000 @ GOTO 2000
1380 IF C(6)<2 THEN 1000 ELSE CLEAR @ GOTO 1320
1500 IF D4$[4,4]="L" THEN D4$[4,4]="0" @ GOTO 1000
1510 CLEAR @ DISP "Name of Blankfile (9 Characters)";@
     INPUT H1$@ D4$[4,4]="L"
1515 IF LEN(H1$)<9 THEN GOTO 1510
1520 H1$=H1$&"    " @ D4$[6,14]=H1$
1540 CREATE D4$[6,14]&".KSYS",5,256 @ ASSIGN# 1 TO
     D4$[6,14]&".KSYS"
1560 FOR J=1 TO 20 @ PRINT# 1 ; T(J) @ NEXT J
1570 FOR J=0 TO T(5) @ PRINT# 1 ; T1(J),T2(J) @ NEXT J
1580 ASSIGN# 1 TO * @ H1$=D4$[6,14]
1590 PRINTER IS 2 @ PRINT "Blank Stored in File: ";H1$
     @ RETURN
1600 CAT "."&B9$ @ DISP "Purge File" @ INPUT H1$@ PURGE
     H1$ @ GOTO 1000
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP "Correct
     Error condition  +CONT";ERRL
1670 PAUSE
1675 GOTO 500
1690 GOSUB 1700 @ GOTO 1000
1700 DISP "Comment 2 Lines" @ FLIP @ INPUT T1$@ FLIP
1710 T1$=T1$&" " @ IF LEN(T1$)<63 THEN GOTO 1710
1720 RETURN
1800 RESET 7 @ ASSIGN# 1 TO D4$[6,14]&".KSYS"
1820 FOR J=1 TO 20 @ READ# 1 ; Z8@ IF J=3 OR J=4 OR J=6
     OR J=7 OR J=15 OR J=16 THEN GOTO 1840
1835 IF Z8#T(J) THEN BEEP @ GOTO 1885
1840 NEXT J @ Z8=1
1850 FOR K=0 TO T(5)
```

```
1860 READ# 1 ; T1(K),T2(K)
1870 T1(K)=-T1(K) @ T2(K)=-T2(K) @ NEXT K @ ASSIGN# 1
     TO *
1875 H1$=D4$[6,14] @ PRINTER IS 2 @ PRINT "Blank:
     ";H1$;" subtracted" @ RETURN
1885 DISP "Variable";J;T(J);"# Blank:";Z8
1890 DISP "Change Variable or (999=ignore)";@ INPUT H1@
     IF H1=999 THEN 1840
1892 IF J=1 OR J=2 OR J=10 OR J=11 OR J=12 OR J=13 OR
     J=19 THEN BEEP @ GOTO 1000 ELSE T(J)=H1
1895 GOTO 1840
1900 A$="*1" @ GOSUB 7000 @ W(11)=.5 @ A$="*A0.5" @
     GOSUB 7000
1905 A$="*B" @ GOSUB 7000 @ WAIT 2000 @ GOSUB 7000 @
     WAIT 2000
1910 MODE 0,1
1920 REFERENCE 10
1930 IF NMEAS=0 THEN GOTO 1930
1940 MEASURE .1
1950 IF NMEAS=0 THEN GOTO 1950
1960 FOR J=1 TO 10 @ IF VALUE(L(J))>5 THEN GOTO 1900
1970 NEXT J @ C(9)=1 @ RETURN
2000 ERASE STATUS
2020 IF T(14)<T(4)/20 OR T(14)>T(4)/3 THEN BEEP @ DISP
     "[S] wrong !" @ WAIT 3000 @ GOTO 500
2040 GOSUB 3600 @ CLEAR @ PRINTER IS 1 @ H1=0 @ GOSUB
     9510
2050 DISP "1=Exit 2=Input"
2200 GOSUB 3650 @ Z5=0
2230 F7$="0" ! TP
2240 IF Q2=2 THEN GOTO 2247
2245 FOR K=0 TO T(5) @ GOTO 2249
2247 FOR K=T(5) TO 0 STEP -1
2249 Z5=Z5-1 @ IF Z5<0 THEN Z5=0
```

```
2250 IF F7$="2" THEN F7$="0" @ K=P3
2255 FOR J=1 TO 80 @ S3(J)=0 @ NEXT J
2260 R9=0 @ R8=0 @ S6=0 @ S7=0
2300 T(0)=INT(T(20)-K*15/T(5)) @ IF K=0 OR T(0)<0 OR
     T(0)>T(20) THEN T(0)=0
2302 T(0)=T(0)+T(8) @ IF T(0)>10 THEN T(0)=10
2305 FOR J=1 TO T(0) @ F6$="0"
2310 GRAPH
2350 GOSUB 3500 ! MESSE
2420 GOSUB 4600 ! DERIV
2430 GOSUB 3300 ! EM TEST
2450 IF F5$="1" THEN GOTO 2310
2470 FOR J1=2 TO T9/L2+1
2480 S3(J1)=S3(J1)+S1(J1)
2490 NEXT J1 @ R9=R9+S7 @ R8=R8+S6 @ IF F6$="1" THEN
     GOTO 2530
2500 NEXT J
2530 GOSUB 3700 ! MITT
2580 IF F6$="1" THEN GOTO 2595
2590 FOR J=2 TO T9/L2+1 @ S3(J)=S3(J)/T(0) @ NEXT J @
     IF F6$="2" THEN GOSUB 4510
2595 F6=3 @ GOSUB 6500
2600 GOSUB 3800 ! CHECK
2620 IF F5$="1" THEN GOSUB 4500
2650 T1(K)=T1(K)+T1(41) @ T2(K)=T2(K)+T2(41) @ IF
     D4$[5,5]="0" THEN 2654
2651 IF ABS(T1(K))>1 OR ABS(T2(K))>1 OR ABS(S9)>1 THEN
     PRINT K;T1(K);T2(K);S9 @ GOTO 2654
2652 PRINTER IS 2 @ PRINT USING "DD,X,SD.6D,X,SD.6D,
     X,D.6D" ; K,T1(K),T2(K),S9
2654 IF F7$="1" THEN F7$="0" @ GOTO 3000
2655 NEXT K
2995 DISP T1$ @ DISP @ DISP @ A$="*WFCC8" @ GOSUB 7000
     @ GOSUB 7700 @ I=BIT(I5,2)
```

```
2997 IF C(6)>1 AND I=0 THEN A$="*5" @ GOSUB 7000 @ WAIT
     1000 @ A$="*I" @ GOSUB 7000
3000 ON KEY# 3,"Rep.Me" GOTO 3100 @ ON KEY# 2,"Output"
     GOTO 3250 @ ON ERROR GOTO 3000
3010 ON KEY# 8,"Syringe" GOTO 3050 @ ON KEY# 4,"Next
     Pr" GOTO 3030 @ ON KEY# 1,"Exit" GOTO 500
3015 ON KEY# 5,"ResM" GOTO 3060 @ ON KEY# 6,"** End-M"
     GOTO 3000 @ ON KEY# 7,"enu **" GOTO 3000
3017 KEY LABEL @ J=0
3019 I5=SPOLL(708) @ I=BIT(I5,3) @ J=J+1 @ BEEP 300,10
3020 WAIT 2000 @ IF C(6)>1 AND J>2 AND I=1 THEN 3250
     ELSE 3019
3030 GOSUB 900 @ J=2 @ GOTO 3019
3050 GOSUB 7250 @ GOTO 3000
3060 F7$="2" @ CLEAR @ DISP "Proceed from Point No." @
     INPUT K@ P3=K @ GOSUB 3110
3070 F5$="0" @ GOTO 2240
3100 F7$="1" @ CLEAR @ DISP "Repeat which Point" @
     INPUT K@ GOSUB 3110 @ GOTO 2255
3110 IF D4$[4,4]#"L" THEN T1(K)=0 @ T2(K)=0 @ GOTO 3200
3120 ASSIGN# 1 TO D4$[6,14]&".KSYS" @ FOR J=1 TO 20 @
     READ# 1 ; T1(41)@ NEXT J
3130 FOR J=0 TO T(5) @ READ# 1 ; Y0,Y1@ IF K=J AND
     F7$="1" THEN T1(K)=-Y0 @ T2(K)=-Y1
3140 IF F7$="2" AND Q2=2 AND K>=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3145 IF F7$="2" AND Q2=1 AND K<=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3150 NEXT J @ ASSIGN# 1 TO *
3200 RETURN
3250 IF F$#"A" OR C(6)<2 THEN 3255
3251 A$="*4" @ GOSUB 7000 @ I=B(20) @ IF E(1)=0 THEN
     H1=W(I+5) @ GOTO 3253
```

```
3252 H1=ABS(W(I+5)/E(1)) @ IF I=2 THEN H1=H1*T(3) ELSE
     H1=H1*T(4)
3253 A$="*"&VAL$(I) @ GOSUB 7000 @ W(I+5)=H1 @
     A$="*="&VAL$(H1) @ GOSUB 7000
3254 A$="*E8" @ GOSUB 7000
3255 IF Q1=2 THEN RESET 7 @ H1$=D4$[15,20]&D1$&VAL$
     (D2)&"N" @ GOSUB 1520 @ D4$[4,4]="L"
3270 GOSUB 4200 @ GOTO 1380
3300 I3=0 @ S7=0 @ F5$="0" @ ON KEY# 2 GOTO 3390 @ BEEP
     100,10 ! EMTES
3310 FOR J1=T8/L2+1 TO T9/L2+1 @ H1=(S9-S1(J1))^2 @
     I3=I3+1 @ S7=S7+H1
3320 NEXT J1 @ S6=(S7/I3)^.5 @ IF S6<.00001*(1+K/T(5))
     *B(2) THEN GOTO 3350
3330 IF K=0 THEN GOTO 3350
3340 IF S6>ABS(S9/100*B(2)) THEN DISP "Meas";K;J;
     "Repeated";Z5 @ BEEP @ F5$="1" @ Z5=Z5+1
3345 IF Z5>3*T(8) AND F5$="1" THEN PRINT J;S9;S8;" ??"
     @ F5$="0"
3350 BEEP 200,10 @ OFF KEY# 2 @ RETURN
3390 I6=1 @ GOSUB 4450 @ GOSUB 4550 @ GOTO 3300
3500 T8=INT((B(13)+K/T(5)*(T(6)-B(13)))/L2)*L2
3503 T9=INT((B(14)+K/T(5)*(T(7)-B(14)))/L2)*L2
3505 IF J#1 THEN GOTO 3510 ELSE I=0 @ GOSUB 7900 @ WAIT
     500
3507 IF K<2 OR K=T(5) OR T(5)<7 THEN A$="*H" @ GOSUB
     7000 @ WAIT 3000+B(9)*K/T(5)
3510 MODE 0,1 @ A$="*H" @ GOSUB 7000 @ WAIT
     B(8)+B(9)*K/T(5)
3512 H4=(T(11)-T(10))/2+1 @ H2=(T(2)-T(1))/2+1 @
     H3=(T(13)-T(12))/2+1
3514 ERASE MEMORY -1 ! MESSE
3515 ON ERROR GOTO 3597
3520 MEASURE L1,L2,0,T9+L2
```

```
3530 FOR J1=1 TO T9/L2+2
3540 IF NMEAS#J1 THEN GOTO 3540
3550 TO MEMORY J1
3560 IF L1>.2 THEN GOSUB 4400
3570 NEXT J1 @ I6=SPOLL(708) @ IF BIT(I6,0)=1 THEN 2995
3575 ALPHA @ IF L2>.2 THEN 3595
3580 FOR J1=1 TO T9/L2+2
3585 RECALL MEMORY J1
3587 GOSUB 4400 @ NEXT J1
3595 RETURN
3597 IF ERRN=2 THEN GOSUB 1900 @ PRINT "Ref. repeated"
     @ GOTO 3500 ELSE GOTO 1660
3600 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
3610
3605 ASSIGN# 1 TO D$ @ DISP "File exists" @ ASSIGN# 1
     TO * @ BEEP @ WAIT 2000 @ GOTO 1000
3610 IF ERRN=130 OR ERRN=125 THEN DISP B9$;"not found"
     @ BEEP @ WAIT 2000 @ GOTO 3600
3620 ON ERROR GOTO 1660 @ PLOTTER 1 @ PRINTER 2
3625 K=T(5) @ GOSUB 8250 @ L(20)=L(20)*T(14)/H4 @
     C(10)=0
3630 B(6)=T(4)/20 @ GOSUB 4800 @ GOSUB 8000 @ IF
     D4$[5,5]="0" THEN H1=0 ELSE H1=1
3635 GOSUB 9500
3640 FOR K=0 TO 40 @ T1(K)=0 @ T2(K)=0 @ NEXT K
3645 IF D4$[4,4]="L" THEN GOSUB 1800
3647 RETURN
3650 F6$="0" @ IF D4$[5.5]="1" THEN PRINTER 2 @ PRINT @
     PRINT "NR   Range 1   Range 2  Noise 1"
3655 DISP "Duration";INT(T(8)*T(5)*(T(7)+5)
     /200+1)*3;"Minutes"
3660 DISP "No.      Range 1      Range 2"
3680 RETURN
3700 T1(41)=0 @ T2(41)=0 @ H4=W1(3)*T(4)/B(7)
```

```
3710 FOR J=1 TO T(0) @ T1(41)=T1(41)+S4(J) @
     T2(41)=T2(41)+S5(J) @ NEXT J
3720 T1(41)=T1(41)/T(0) @ T2(41)=T2(41)/T(0) @
     R9=R9/T(0) @ R9=R9^.5 @ R8=(R8/T(0))^.5
3730 DISP USING "DD,X,5D.DD,XXX,DDD.DD,XXX,.6D" ;
     K,H4,B(3),R8 @ RETURN
3800 S9=0 @ S8=0 @ F5$="0" ! CH
3810 FOR J=1 TO T(0)
3820 S9=S9+(T1(41)-S4(J))^2 @ S8=S8+(T2(41)-S5(J))^2 @
     NEXT J @ S9=(S9/T(0))^.5
3830 IF S9<ABS(.00005*B(1)) OR Q1#1 THEN GOTO 3860
3850 IF S9>ABS(T1(41)/100*B(1)) THEN DISP "Average";K;"
     repeated";Z5 @ BEEP @ F5$="1"
3855 Z5=Z5+T(0) @ IF Z5>2*T(8) AND F5$="1" THEN 3870
3860 RETURN
3870 PRINT "Average nonreliable" @ FOR J=1 TO T(0) @
     PRINT "Msg";K;J;" ";S4(J);S5(J) @ NEXT J
3875 F5$="0" @ GOTO 3860
4200 IF D4$[5,5]="0" THEN 4205
4201 PRINTER 2 @ PRINT @ PRINT @ PRINT TAB(7);E4$ @
     GRAPH @ COPY
4202 H1=0 @ GOSUB 9500 @ PRINT T1$ @ ON ERROR GOTO 1660
4205 PRINT @ PRINT @ PRINT "No.  Range 1   Range 2
     [Substr]"
4210 PRINT "    TN(1/sec) TN(1/sec)    M" @ PRINT
4220 K=T(5) @ GOSUB 8250
4230 A$="PFTN.KSYS"
4250 GOSUB 8500
4260 LORG 0 @ CSIZE 3 @ GOSUB 5500
4270 FOR K=0 TO T(5) @ GOSUB 8250
4290 MOVE H4,H3 @ LABEL "*" @ MOVE H4,J1 @ LABEL "+"
4310 PRINTER IS 2 @ PRINT USING "DD,X,S5D.D,X,
     S5D.D,XX,5D.DD" ; K,H3,J1,H4
4312 PRINT# 1 ; H3,J1,H4
```

```
4320 NEXT K @ PRINTER IS 2
4330 PRINT @ IF D4$[5,5]="1" THEN PRINT @ PRINT
TAB(7);E4$ @ COPY @ PRINT @ PRINT
4350 ASSIGN# 1 TO *
4360 GOSUB 6000 @ PRINT @ PRINT @ PRINT @ RETURN
4400 I3=0 @ H1=0
4410 FOR I2=T(1) TO T(2) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4415 S1(J1)=H1/H2 @ H1=0
4420 FOR I2=T(12) TO T(13) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4425 S2(J1)=H1/H3 @ H1=0
4430 FOR I2=T(10) TO T(11) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4435 S1(J1)=S1(J1)-H1/H4 @ S2(J1)=S2(J1)-H1/H4
4440 RETURN
4450 DISP "1=Exit 2=New Stdev 3=Rep.Single" @ DISP
     "4=Rep.Aver. 5=End Me  6=go on"
4455 DISP "7=Test Msg 8=Next Ready";
4460 INPUT H1@ RETURN
4500 IF F$="A" THEN GOTO 2250
4510 GOSUB 4450 @ ON ERROR GOTO 1660 @ I6=2
4520 F6$="0" @ IF H1#3 THEN 4550
4530 F6$="1" @ DISP "No. of Single Measurement" @ INPUT
     J@ FOR J1=1 TO 80 @ S3(J1)=0 @ NEXT J1
4540 R9=R9^2*T(0)/(1.2+T(0)) @ R8=R8^2*T(0)/(1.2+T(0))
     @ GOTO 2310
4550 IF H1=5 THEN 2995
4555 IF H1=7 THEN 2000
4560 IF H1=2 THEN DISP "Max. Nonlin./Noise 2 Inputs";@
     INPUT B(1),B(2)@ GOTO 4590
4565 IF H1=8 THEN GOSUB 900 @ GOTO 4590
4570 IF H1=3 AND I6=1 THEN F6$="2" @ GOTO 4590
4580 IF H1=4 THEN 2250
```

```
4590 RETURN
4600 S9=0 @ S8=0 @ I1=0
4620 H1=S1(1) @ H2=S2(1)
4630 FOR J1=2 TO T9/L2+1
4635 H3=S1(J1) @ H4=S2(J1)
4640 S1(J1)=(S1(J1+1)-H1)/(2*L2) @ S2(J1)=(S2(J1+1)-
     H2)/(2*L2) @ IF J1>T9/L2+1 THEN GOTO 4710
4700 IF J1>=T8/L2+1 THEN I1=I1+1 @ S9=S9+S1(J1) @
     S8=S8+S2(J1)
4710 H1=H3 @ H2=H4 @ NEXT J1 @ I3=0 @ S8=S8/I1 @
     S9=S9/I1
4730 DISP USING "DD,X,DD,XX,SZ.6D,XXX,SZ.6D" ;
     K,J,S9,S8
4740 S4(J)=S9 @ S5(J)=S8
4750 RETURN
4800 F6$="0" @ A5=0 @ A6=0 @ MODE 0,1
4810 LAMBDA L(1),L(2),L(3),L(4),L(5),L(6),L(7),
     L(8),L(9),L(10)
4820 TIME SCALE 0 TO T(7)
4830 ABSORBANCE
4832 IF B(3)>T(3)/3 THEN B(3)=T(3)/3
4833 IF B(6)>T(4)/3 THEN B(6)=T(4)/3
4835 W1(3)=B(7)*B(6)/T(4)
4836 W1(2)=B(7)*B(3)/T(3)
4840 IF W1(3)<=0 THEN GOTO 1900
4845 IF C(9)#1 THEN 1000
4850 I=0 @ GOSUB 7980
4860 A$="*H" @ GOSUB 7000
4865 WAIT 5000
4870 J=0 @ K=INT(T(5)/2) @ T9=T(7) @ T8=B(13) @ GOSUB
     3510
4920 H1=0 @ H2=0
4930 FOR J1=1 TO 4 @ H1=S1(J1)+H1 @ H2=S2(J1)+H2 @ NEXT
     J1 @ H1=H1/4 @ H2=H2/4
```

```
5050 IF D4$[5,5]#"0" AND A5#1 THEN A$="PFOD.KSYS" @
     GOSUB 8500 @ A5=1
5055 DISP "Initial OD Range 1 and Range 2"
5056 DISP USING "6X,SD.5D,6X,SD.5D" ; H1,H2 @
     H1=H1/T(15)*1000 @ H2=H2/T(16)*1000
5057 IF ABS(H1)>99999 OR ABS(H2)>99999 THEN DISP
     "[E]";H1;H2;"uM" @ GOTO 5060
5058 DISP USING "3A,X,S5D.3D,4X,S5D.3D,2A" ;"[E]",H1,
     H2,"uM" @ DISP "Deriv. Range1 and Range 2"
5060 IF D4$[5,5]#"0" THEN F6=1 @ GOSUB 6500 @ F6=2 @
     GOSUB 6500
5150 K=0 @ J=0 @ GOSUB 4600 @ GOSUB 3300 @
     H4=ABS(S6/S9*100) @ IF H4>999 THEN H4=999
5160 DISP USING "14A,X,.6D,3A,3D.D,A" ; "Noise  Range
     1",ABS(S6)," = ",H4,"%" @ H1=0 @ H2=0
5170 FOR J=2 TO 6 @ H1=H1+S1(J) @ NEXT J @ H1=H1/5 @
     A1=H4
5180 FOR J=T(7)/L2-3 TO T(7)/L2+1 @ H2=H2+S1(J) @ NEXT
     J @ H2=H2/5 @ H3=INT(ABS((H1-H2)/H1*100))
5190 DISP "Nonlinearity  : ";H3;"%" @ A2=H3 @ DISP @
     DISP
5300 J1=0 @ IF D4$[5,5]#"0" THEN GOSUB 5400
5310 IF J1=9 THEN 4810
5350 IF A5#2 THEN A$="PFAB.KSYS" @ GOSUB 8500 @ A5=2
5360 F6=1 @ GOSUB 6500 @ F6=2 @ GOSUB 6500
5390 GOSUB 5400 @ IF J1=9 THEN 4810
5395 RETURN
5400 OFF KEY# 8 @ ON KEY# 2,"Go on" GOTO 5490 @ ALPHA
5402 ON KEY# 5,"[E]" GOTO 5440 @ ON KEY# 6,"Activ" GOTO
5445 @ OFF KEY# 7
5405 ON KEY# 3,"Graph" GOTO 5420 @ ON KEY# 4,"Alpha"
     GOTO 5430 @ KEY LABEL
5407 IF F$="A" AND C(10)=0 THEN J1=1 ELSE J1=0
```

```
5410 IF J1=0 THEN 5410 ELSE BEEP 300,20 @ WAIT 3000 @
     GOTO 5450
5420 GRAPH @ J1=0 @ GOTO 5410
5430 ALPHA @ J1=0 @ GOTO 5410
5440 DISP B(3);"new";@ INPUT B(3)@ B(5)=B(3) @ J1=9 @
     GOTO 5490
5445 DISP B(10);"new";@ INPUT B(10)@ GOTO 5400
5450 IF B(4)=0 OR Q1=2 THEN 5490
5454 IF B(3)<T(3)/200 THEN B(3)=T(3)/200 @ GOTO 5490
5455 IF ABS(S9)>2*B(10) THEN H1=.5 @ GOTO 5492
5460 IF ABS(S9)<B(10)/2 THEN H1=2 @ GOTO 5492
5465 IF ABS(S9)>1.3*B(10) THEN H1=.75 @ GOTO 5492
5470 IF ABS(S9)<.75*B(10) THEN H1=1.25 @ GOTO 5492
5490 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ RETURN
5492 B(3)=B(3)*H1 @ A6=A6+1 @ CLEAR @ DISP
     "Enzymeconc.:";B(3) @ IF A6>6 THEN J1=0 ELSE J1=9
5493 GOTO 5490
5495 GOSUB 7250 @ GOTO 5400
5500 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     1660
5550 CREATE D$,3+INT(T(5)/10+.5),256
5560 ASSIGN# 1 TO D$
5570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T(),L(),B()
5660 RETURN
6000 A1=0 @ A2=0 @ A3=0 @ A4=0 @ A5=0 @ A6=0
6050 FOR K=2 TO T(5) @ GOSUB 8250
6090 IF H3=0 OR H4=0 THEN GOTO 6200 ELSE H1=1/H4 @
     H2=1/H3
6120 A1=A1+1 @ A2=A2+H1 @ A3=A3+H2 @ A4=A4+H1*H2 @
     A5=A5+H1*H1 @ A6=A6+H2*H2
6200 NEXT K
6210 IF A1=0 THEN GOTO 6400
6220 H1=A5/A1-(A2/A1)^2
```

```
6230 H2=A6/A1-(A3/A1)^2
6240 H3=(A4/A1-A2/A1*A3/A1)/H1
6250 H4=A3/A1-H3*A2/A1
6260 H5=H3*H1^.5/H2^.5
6280 H1=ABS(1/(H4/H3)) @ H2=ABS(1/H4)
6290 IF ABS(H1)>9999 OR ABS(H2)>99999 OR ABS(H5)>1 THEN
     GOTO 6400
6300 PRINT USING "15A,4X,6D.2D,X,3A" ; "Velocity
     =",H2,"1/s"
6310 PRINT USING "16A,3X,5D.3D,X,3A" ; "Michaelis
     Const.=",H1," M"
6320 PRINT USING "14A,9X,3D.4D" ; "Correlation   =",H5
6400 PRINT @ PRINT @ RETURN
6500 PLOT 0,0,2
6520 FOR J1=2 TO T9/L2+1
6530 H4=(J1-1)*L2
6540 ON F6 GOTO 6550,6560,6570
6550 H3=S1(J1) @ GOTO 6580
6560 H3=S2(J1) @ GOTO 6580
6570 H3=S3(J1)
6580 PLOT H4,H3,1
6590 NEXT J1 @ PENUP @ PLOT 0,0,2 @ RETURN
7000 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
7005 SET TIMEOUT 7;3000
7010 ON TIMEOUT 7 GOTO 7090
7015 A1$=A$[2,2]
7020 I5=SPOLL(708)
7025 IF BIT(I5,7) THEN WAIT 500 @ GOTO 7020
7030 IF BIT(I5,5) THEN DISP "Empty" @ F5$="2" @ C(15)=0
     @ GOTO 7082
7032 IF BIT(I5,4) THEN DISP "Full" @ C(15)=0
7035 IF BIT(I5,3) THEN C(15)=0
7040 A$=A$&"," @ OUTPUT 708 ;A$
```

```
7052 IF A1$="1" THEN M=1
7054 IF A1$="2" THEN M=2
7060 IF A1$="3" THEN M=3
7061 IF A1$="B" OR A1$="D" THEN C(M)=C(M)-W(M+10)
7062 IF A1$="C" THEN C(M)=C(M)+W(M+10)
7063 IF A1$="F" THEN C(M)=0
7064 IF A1$="G" THEN C(M)=W(M)
7065 IF A1$="H" THEN C(1)=C(1)-W1(1) @ C(2)=C(2)-W1(2)
     @ C(3)=C(3)-W1(3)
7070 OFF TIMEOUT 7
7080 RETURN
7082 IF A1$="F" OR A1$="B" OR A1$="D" THEN 7080 ELSE
7040
7090 DISP "Switch on ASSAYOMAT" @ BEEP @ WAIT 5000 @
     RESET 7 @ GOTO 7000
7100 DISP A$ @ A$="*TF84A " @ I6=W1(4) @ GOSUB 7200
7110 I6=INT(W1(4)/(W1(1)*48000/W(1)))+257 @ GOSUB 7200
7115 IF W1(2)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(2)*48000/W(2)))+257
7120 GOSUB 7200 @ IF W1(3)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(3)*48000/W(3)))+257
7125 GOSUB 7200
7130 GOSUB 7000
7190 RETURN
7200 B$="    " @ H1$="0123456789ABCDEF"
7205 FOR J1=4 TO 1 STEP -1
7210 I=I6 MOD 16 @ B$[J1,J1]=H1$[I+1,I+1] @ I6=I6-I @
     I6=I6/16 @ NEXT J1
7215 A$=A$&"="&B$[3,4]&"="&B$[1,2]
7220 RETURN
7250 M=0 @ A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700 @ GOTO
     7480
7280 ON KEY# 1,"Give" GOTO 7430 @ ON KEY# 2,"Suck" GOTO
     7420
```

```
7300 ON KEY# 3,"Back" GOTO 7400 @ ON KEY# 4,"Empty"
     GOTO 7460
7320 ON KEY# 5,VAL$(W(M+10)) GOTO 7360
7330 ON KEY# 6,VAL$(C(M)) GOTO 7440
7340 ON KEY# 7,"Retur" GOTO 7355
7350 ON KEY# 8,"Syr."&VAL$(M) GOTO 7480
7352 KEY LABEL @ GOTO 7280
7355 OFF KEY# 1 @ RETURN
7360 CLEAR @ DISP "Volume per Key";@ INPUT H1$@
     W(M+10)=VAL(H1$) @ A$="*A"&H1$ @ GOSUB 7000
7370 GOTO 7280
7400 A$="*D" @ GOSUB 7000 @ GOTO 7280
7420 A$="*C" @ GOSUB 7000 @ GOTO 7280
7430 A$="*B" @ GOSUB 7000 @ GOTO 7280
7440 A$="*G" @ GOSUB 7000 @ GOTO 7280
7460 A$="*F" @ GOSUB 7000 @ GOTO 7280
7480 M=M+1 @ IF M>3 THEN M=1
7485 A$="*"&VAL$(M) @ GOSUB 7000 @ GOTO 7280
7500 CLEAR @ DISP "No. of Cycles and Syringe (1-3)";@
     INPUT H1,M
7510 IF M<1 OR M>3 THEN 7500
7520 A$="*"&VAL$(M) @ GOSUB 7000
7530 A$="*E"&VAL$(H1) @ GOSUB 7000
7535 IF H1 MOD 2#0 THEN C(M)=0 ELSE C(M)=W(M+5)
7540 RETURN
7700 SET TIMEOUT 7;2000 @ A$=""
7710 ON TIMEOUT 7 GOTO 7790
7720 ENTER 708 USING "#,B" ; H1@ A$=A$&CHR$(H1) @ IF
     H1=13 OR LEN(A$)>40 THEN 7740
7730 GOTO 7720
7740 OFF TIMEOUT 7 @ ON ERROR GOTO 7790
7750 C(1)=VAL(A$[10,16])
7760 C(2)=VAL(A$[20,26]) @ C(3)=VAL(A$[30,36]) @
     C(15)=1
```

```
7790 ON ERROR GOTO 1600 @ RETURN
7900 IF K=0 THEN N3=0 @ GOTO 7970
7910 ON T(19) GOTO 7920,7930,7950,7940,7960
7920 N3=4*L3^(K-1) @ GOTO 7970
7930 N3=K*300/T(5) @ IF K>T(5)/2 THEN N3=150+INT(K-
T(5)/2)*600/T(5) @ GOTO 7970 ELSE GOTO 7970
7940 K1=T(5)+1-K @ N3=1/K1*600 @ GOTO 7970
7950 N3=5+K*100/T(5)+3*L3^(K-1) @ GOTO 7970
7960 N3=N3/L(20)
7970 W1(3)=N3*L(20)/10000
7975 W1(2)=B(7)*B(3)/T(3)*(1-B(17)*(T(5)-K)/T(5))
7980 W1(1)=B(7)-W1(2)-W1(3) @ IF I=9 THEN RETURN
7985 A$="*K"&VAL$(W1(1))&","&VAL$(W1(2))&",
     "&VAL$(W1(3)) @ GOSUB 7100
7990 GOSUB 7000 @ RETURN
8000 S1(1)=4 @ S1(2)=.3 @ S1(3)=.25 @ I=9 @ A$="*WFCC8"
     @ GOSUB 7000 @ GOSUB 7700
8010 FOR K=0 TO T(5) @ GOSUB 7900 @ FOR M=1 TO 3 @
     S1(M)=S1(M)+W1(M)*T(8)*1.1 @ NEXT M @ NEXT K
8060 FOR M=1 TO 3 @ IF S1(M)>C(M) THEN DISP "Syringe
     ";M;" is filled" @ GOSUB 8100
8070 NEXT M @ RETURN
8100 K=B(20) @ A$="*"&VAL$(M) @ GOSUB 7000 @ IF M#K
     THEN 8120
8105 S1(K)=S1(K)+.2 @ IF S1(K)>W(K) THEN S1(K)=W(K)
8107 IF S1(K)>B(19) THEN S1(K)=B(19)
8110 A$="*="&VAL$(S1(K)) @ GOSUB 7000 @ W(K+5)=S1(K)
8120 A$="*G" @ GOSUB 7000
8190 RETURN
8200 I=0 ! BER. [E] [S]
8205 GOSUB 7900
8210 H2=T(3)*W1(2)/B(7)
8220 H4=T(4)*W1(3)/B(7) ! [S]
```

```
8230 H3=T1(K)*1000000/T(17)/H2 @ J1=T2(K)*1000000
     /T(18)/H2
8240 RETURN
8250 I=9 @ GOSUB 8205
8260 RETURN
8500 RESET 7
8510 ASSIGN# 1 TO A$ @ READ# 1 ; H3,H4,X0,X1,X2,X3,
     Y0,Y1,Y2,Y3,E3$,E4$
8550 GRAPH @ GCLEAR @ SCALE 0,100,0,100 @ IF X2=0 THEN
     X2=(X1-X0)/5
8560 MOVE 30,0 @ LDIR 0 @ CSIZE 3 @ LABEL E3$ @ F5=(X1-
     X0)/7 @ F6=(Y1-Y0)/10
8590 SCALE X0-F5,X1,Y0-F6,Y1 @ XAXIS Y1,X3,X0,X0+X3 @
     XAXIS Y1,X2,X0+X3,X1
8620 XAXIS Y0,X3,X0,X0+X3 @ XAXIS Y0,X2,X0+X3,X1 @
     YAXIS X1,Y3,Y0,Y0+Y3
8650 YAXIS X1,Y2,Y0+Y3,Y1 @ YAXIS X0,Y3,Y0,Y0+Y3 @
     YAXIS X0,Y2,Y0+Y3,Y1
8660 FOR K=X0+X3 TO X1 STEP X2
8670 MOVE K-F5/5,Y0-(Y1-Y0)/20
8680 LABEL VAL$(K) @ NEXT K
8690 FOR K=Y0+Y3 TO Y1 STEP Y2
8700 MOVE X0-(X1-X0)/10,K-F6/2
8710 LABEL VAL$(K) @ NEXT K
8900 ASSIGN# 1 TO * @ RETURN
9500 PRINT @ PRINT @ PRINT
9510 PRINT "Measurement ";D4$[15,20];"  ";D1$;D2 @
     PRINT "Diskette:";B9$;"  ";D3$
9540 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9542 IF H1=0 THEN 9700
9545 PRINT "Time    [S]min";TAB(16);B(13);
     "bis";TAB(24);B(14);"sec"
9546 PRINT "Time    [S]max";TAB(16);T(6);"bis";
     TAB(24);T(7);"sec"
```

```
9547 PRINT "Interval  ";L1;" / Integration";L2
9548 PRINT "Max.Stdev";B(1);" / Nonlin.";B(2);"%"
9550 PRINT "Range 1 from   ";T(1);" to";T(2);"nm"
9555 PRINT "Range 2 from   ";T(12);" to";T(13);"nm"
9557 PRINT "Int. Ref. from ";T(10);" to";T(11);"nm"
9560 PRINT "[";E1$;"]";TAB(23);T(3);"nM"
9565 PRINT "in the Assay :  ";B(17);TAB(23);B(3);"nM"
9580 PRINT "[";E2$;"]";TAB(23);T(4);"M"
9585 PRINT "Max. Conc. in Assay:  ";T(14);"M"
9600 PRINT "No. of Points";T(5);TAB(24);"Rep.";T(8)
9650 PRINT "Delta Epsilon 1 :  ";T(17);"1/mM cm" @
     PRINT "Delta Epsilon 2 :  ";T(18);"1/mM cm"
9657 PRINT "Assay Volume :";TAB(25);B(7);"ml"
9690 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9700 PRINTER IS 2 @ RETURN
```

APPENDIX K
Measure Overlay KININ

```
1 ! KININ B.MICHEL 02.02.88
120 COM SHORT S1(80),S2(80),S3(80),T(20),T1(41),T2(41),
    L(20),B(20),A(5,5)
125 COM SHORT C(15),W(20),W1(5),E(20)
140 COM T1$[65],E1$[20],E2$[20],D$[20],D1$[1],D2,
    D3$[20],D4$[20],B9$[8]
180 DIM F$[1],F6$[1],E3$[30],E4$[30],F7$[1],F5$[1],
    A$[60],H1$[20]
195 INTEGER I,I2,I5,I6
200 F6$="0" @ Q1=1 @ Q2=1 @ F7$="0" @ CLEAR @ I=0
210 F$="A" @ F5$="0" @ J=1 @ K=1 @ D4$[5,5]="1"
250 ON ERROR GOTO 800 @ PLOTTER 1 @ IF C(8)#1 THEN GOTO
    800
500 L1=B(11) @ L2=B(12) @ L3=256^(1/T(5)) @ ON ERROR
    GOTO 500
510 CLEAR @ DISP " Inhibition Screening Menu" @ DISP
520 DISP E1$;TAB(19);T(3);"nm (E)" @ DISP
    E2$;TAB(19);T(4);"m (S)" @ DISP
530 DISP "No. of Rep.";T(8) @ DISP "No. of Points
    ";T(5)
650 ON KEY# 1,"DiVar" GOTO 720 @ ON KEY# 2,"MeasM" GOTO
    1000 @ ON KEY# 3,"Syringe" GOTO 860
655 ON KEY# 4,"Wash" GOTO 700 @ ON KEY# 5,"MAINPG" GOTO
800 @ ON KEY# 6,"TestM" GOTO 790
660 ON KEY# 7,"[E,I]" GOTO 880 @ ON KEY# 8,"Points"
    GOTO 940 @ KEY LABEL @ GOTO 650
700 GOSUB 7500 @ GOTO 500
720 PRINTER IS 1 @ GOSUB 9545 @ PAUSE
725 GOTO 500
760 CLEAR @ IF B(4)=0 THEN B(4)=1 @ DISP "Enter v in
    assay";@ INPUT B(10)@ GOTO 765
```

```
762 B(4)=0 @ DISP "Enter [E] in Assay";@ INPUT B(5)
765 B(3)=B(5) @ GOTO 1000
790 RESET 7 @ CHAIN "KONZMSG.KSYS"
800 RESET 7 @ CHAIN "Autost.KSYS"
820 CLEAR @ DISP "** Single Meas **"
825 DISP E2$ @ DISP "Conc in M at [I]<0: Reference";
830 INPUT B(6)@ C(10)=1
835 DISP E1$ @ DISP "Conc in nM (Stock=";T(3);"nM)";@
     INPUT B(5)
840 B(3)=B(5) @ MODE 0,1 @ GOSUB 4800 @ GOTO 1000
860 GOSUB 7250 @ GOTO 500
880 CLEAR @ DISP E1$;" Conc. in (nM)"
885 DISP "and ";E2$;" Conc. in (M)" @ INPUT T(3),T(4)
890 GOTO 500
900 CLEAR @ IF B(20)=2 THEN DISP E1$;ELSE DISP E2$;
905 DISP " Conc. Nr.";C(6)
910 DISP "in Syringe (C=clear Buffer)" @ DISP "negative
     [E] = decreasing [I]" @ INPUT H1$
915 IF H1$="C" THEN C(6)=1 @ GOTO 900
920 E(C(6))=VAL(H1$) @ C(6)=C(6)+1 @ RETURN
940 CLEAR @ DISP "No. of Points (2-40) and Rep. (2-
     10)";
950 INPUT T(5),T(8)@ IF T(5)>40 OR T(5)<2 OR T(8)>10 OR
     T(8)<0 THEN GOTO 940 ELSE GOTO 500
1000 CLEAR @ DISP "Inhibition Scr Menu" @ DISP @ DISP
     T1$[1,63] @ DISP @ ON ERROR GOTO 500
1020 DISP "Syr.1:";INT(C(1)*10)/10;"2:";INT(C(2)
     *100)/100;"3:";INT(C(3)*100)/100;"ml" @ DISP
1050 DISP "Max. [I]  :  ";T(14);"uM"
1090 IF D4$[4,4]="L" THEN DISP "Blank ";D4$[6,14];"
     subtracted" ELSE DISP "No Blank"
1120 IF B(4)=1 THEN DISP "[I] Autom."
1130 IF D4$[5,5]="1" THEN DISP "Print On" ELSE DISP
     "Print OFF"
```

```
1140 IF C(9)=0 THEN DISP "No Reference" @ BEEP
1150 DISP "Assay Volume : ";B(7);"ul" @ ON ERROR GOTO
     1000
1200 ON KEY# 1,"Exit" GOTO 500 @ ON KEY# 2,"Print" GOTO
1280 @ ON KEY# 3,"Content" GOTO 1600
1230 ON KEY# 4,"Comment" GOTO 1690 @ ON KEY# 5,"KiMeas"
     GOTO 1300 @ ON KEY# 6,"Blank" GOTO 1500
1260 ON KEY# 7,"[I]auto" GOTO 760 @ ON KEY#
     8,"SingleMe" GOTO 820 @ KEY LABEL @ GOTO 1200
1280 IF D4$[5,5]="0" THEN D4$[5,5]="1" ELSE
     D4$[5,5]="0"
1290 GOTO 1000
1300 ON ERROR GOTO 1000 @ IF C(9)=0 THEN GOTO 1000
1302 CLEAR @ DISP "Series and Number eg A,1";@ INPUT
     D1$,B1@ IF B1>999 OR B1<0 THEN 1300
1304 DISP "Memocode (6 Characters)";@ INPUT D4$[15,20]@
     A$="*4" @ GOSUB 7000
1310 M=B(20) @ IF C(M)<W(M)/8 THEN H1=7 @ GOSUB 7510
1320 D2=B1 @ B1=B1+1 @ F8$="N"
1325 IF C(6)=1 THEN GOSUB 900 ELSE H1=0 @ PRINTER IS 1
     @ GOSUB 9500
1330 B(3)=B(5) @ IF E(1)#0 THEN Q1=1 @ GOTO 1335
1332 E(1)=1 @ B(3)=.1 @ Q1=2 @ D4$[4,4]="0" @
     D1$=CHR$(NUM(D1$)+1) @ D2=0
1335 IF E(1)<0 THEN Q2=2 @ E(1)=-E(1) ELSE Q2=1
1337 IF B(20)=2 THEN T(3)=E(1) ELSE T(4)=E(1)
1340 FOR I=1 TO C(6)-1 @ E(I)=E(I+1) @ NEXT I @
     C(6)=C(6)-1
1350 W1(2)=B(7)*B(5)/T(3) @ GOSUB 8000 @ GOTO 2000
1380 IF C(6)<2 THEN 1000 ELSE CLEAR @ GOTO 1320
1500 IF D4$[4,4]="L" THEN D4$[4,4]="0" @ GOTO 1000
1510 CLEAR @ DISP "Name of Blankfile (9 Characters)";@
     INPUT H1$@ D4$[4,4]="L"
1515 IF LEN(H1$)<9 THEN GOTO 1510
```

```
1520 H1$=H1$&"    " @ D4$[6,14]=H1$
1540 CREATE D4$[6,14]&".KSYS",5,256 @ ASSIGN# 1 TO
     D4$[6,14]&".KSYS"
1560 FOR J=1 TO 20 @ PRINT# 1 ; T(J) @ NEXT J
1570 FOR J=0 TO T(5) @ PRINT# 1 ; T1(J),T2(J) @ NEXT J
1580 ASSIGN# 1 TO * @ H1$=D4$[6,14]
1590 PRINTER IS 2 @ PRINT "Blank Stored in File: ";H1$
     @ RETURN
1600 CAT "."&B9$ @ DISP "Purge File" @ INPUT H1$@ PURGE
     H1$ @ GOTO 1000
1660 BEEP 50,1000 @ DISP "Error";ERRN @ DISP "Correct
     Error condition  +CONT";ERRL
1670 PAUSE
1675 GOTO 500
1690 GOSUB 1700 @ GOTO 1000
1700 DISP "Comment 2 Lines" @ FLIP @ INPUT T1$@ FLIP
1710 T1$=T1$&" " @ IF LEN(T1$)<63 THEN GOTO 1710
1720 RETURN
1800 RESET 7 @ ASSIGN# 1 TO D4$[6,14]&".KSYS"
1820 FOR J=1 TO 20 @ READ# 1 ; Z8@ IF J=3 OR J=4 OR J=6
     OR J=7 OR J=15 OR J=16 THEN GOTO 1840
1835 IF Z8#T(J) THEN BEEP @ GOTO 1885
1840 NEXT J @ Z8=1
1850 FOR K=0 TO T(5)
1860 READ# 1 ; T1(K),T2(K)
1870 T1(K)=-T1(K) @ T2(K)=-T2(K) @ NEXT K @ ASSIGN# 1
     TO *
1875 H1$=D4$[6,14] @ PRINTER IS 2 @ PRINT "Blank:
     ";H1$;" subtracted" @ RETURN
1885 DISP "Variable";J;T(J);"# Blank:";Z8
1890 DISP "Change Variable or (999=ignore)";@ INPUT H1@
     IF H1=999 THEN 1840
1892 IF J=1 OR J=2 OR J=10 OR J=11 OR J=12 OR J=13 OR
     J=19 THEN BEEP @ GOTO 1000 ELSE T(J)=H1
```

```
1895 GOTO 1840
1900 A$="*1" @ GOSUB 7000 @ W(11)=.5 @ A$="*A0.5" @
     GOSUB 7000
1905 A$="*B" @ GOSUB 7000 @ WAIT 2000 @ GOSUB 7000 @
     WAIT 2000
1910 MODE 0,1
1920 REFERENCE 10
1930 IF NMEAS=0 THEN GOTO 1930
1940 MEASURE .1
1950 IF NMEAS=0 THEN GOTO 1950
1960 FOR J=1 TO 10 @ IF VALUE(L(J))>5 THEN GOTO 1900
1970 NEXT J @ C(9)=1 @ RETURN
2000 ERASE STATUS
2040 GOSUB 3600 @ CLEAR @ PRINTER IS 1 @ H1=0 @ GOSUB
     9510
2050 DISP "1=Exit 2=Input"
2200 Z5=0
2210 F6$="0" @ IF D4$[5,5]="1" THEN PRINTER 2 @ PRINT @
     PRINT "NR   Range 1   Range 2  Noise 1"
2230 F7$="0" ! TP
2240 IF Q2=2 THEN GOTO 2247
2245 FOR K=0 TO T(5) @ GOTO 2249
2247 FOR K=T(5) TO 0 STEP -1
2249 Z5=Z5-1 @ IF Z5<0 THEN Z5=0
2250 IF F7$="2" THEN F7$="0" @ K=P3
2255 FOR J=1 TO 80 @ S3(J)=0 @ NEXT J
2260 R9=0 @ R8=0 @ S6=0 @ S7=0
2300 T(0)=INT(T(20)-K*15/T(5)) @ IF K=0 OR T(0)<0 OR
     T(0)>T(20) THEN T(0)=0
2302 T(0)=T(0)+T(8) @ IF T(0)>10 THEN T(0)=10
2305 FOR J=1 TO T(0) @ F6$="0"
2310 GRAPH
2350 GOSUB 3500 ! MESSE
2420 GOSUB 4600 ! DERIV
```

```
2430 GOSUB 3300 ! EM TEST
2450 IF F5$="1" THEN GOTO 2310
2470 FOR J1=2 TO T9/L2+1
2480 S3(J1)=S3(J1)+S1(J1)
2490 NEXT J1 @ R9=R9+S7 @ R8=R8+S6 @ IF F6$="1" THEN
     GOTO 2530
2500 NEXT J
2530 GOSUB 3700 ! MITT
2580 IF F6$="1" THEN GOTO 2595
2590 FOR J=2 TO T9/L2+1 @ S3(J)=S3(J)/T(0) @ NEXT J @
     IF F6$="2" THEN GOSUB 4510
2595 F6=3 @ GOSUB 6500
2600 GOSUB 3800 ! CHECK
2620 IF F5$="1" THEN GOSUB 4500
2650 T1(K)=T1(K)+T1(41) @ T2(K)=T2(K)+T2(41) @ IF
     D4$[5,5]="0" THEN 2654
2651 IF ABS(T1(K))>1 OR ABS(T2(K))>1 OR ABS(S9)>1 THEN
     PRINT K;T1(K);T2(K);S9 @ GOTO 2654
2652 PRINTER IS 2 @ PRINT USING "DD,X,SD.6D,X,SD.6D,X,
     D.6D" ; K,T1(K),T2(K),S9
2654 IF F7$="1" THEN F7$="0" @ GOTO 3000
2655 NEXT K
2995 DISP T1$ @ DISP @ DISP @ A$="*WFCC8" @ GOSUB 7000
     @ GOSUB 7700 @ I=BIT(I5,2)
2997 IF C(6)>1 AND I=0 THEN A$="*5" @ GOSUB 7000 @ WAIT
     1000 @ A$="*I" @ GOSUB 7000
3000 ON KEY# 3,"Rep.Me" GOTO 3100 @ ON KEY# 2,"Output"
     GOTO 3250 @ ON ERROR GOTO 3000
3010 ON KEY# 8,"Syringe" GOTO 3050 @ ON KEY# 4,"Next
     Pr" GOTO 3030 @ ON KEY# 1,"Exit" GOTO 500
3015 ON KEY# 5,"ResM" GOTO 3060 @ ON KEY# 6,"** End-M"
     GOTO 3000 @ ON KEY# 7,"enu **" GOTO 3000
3017 KEY LABEL @ J=0
3019 I5=SPOLL(708) @ I=BIT(I5,3) @ J=J+1 @ BEEP 300,10
```

```
3020 WAIT 2000 @ IF C(6)>1 AND J>2 AND I=1 THEN 3250
     ELSE 3019
3030 GOSUB 900 @ J=2 @ GOTO 3019
3050 GOSUB 7250 @ GOTO 3000
3060 F7$="2" @ CLEAR @ DISP "Proceed from Point No." @
     INPUT K@ P3=K @ GOSUB 3110
3070 F5$="0" @ GOTO 2240
3100 F7$="1" @ CLEAR @ DISP "Repeat which Point" @
     INPUT K@ GOSUB 3110 @ GOTO 2255
3110 IF D4$[4,4]#"L" THEN T1(K)=0 @ T2(K)=0 @ GOTO 3200
3120 ASSIGN# 1 TO D4$[6,14]&".KSYS" @ FOR J=1 TO 20 @
     READ# 1 ; T1(41)@ NEXT J
3130 FOR J=0 TO T(5) @ READ# 1 ; Y0,Y1@ IF K=J AND
     F7$="1" THEN T1(K)=-Y0 @ T2(K)=-Y1
3140 IF F7$="2" AND Q2=2 AND K>=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3145 IF F7$="2" AND Q2=1 AND K<=J THEN T1(J)=-Y0 @
     T2(J)=-Y1
3150 NEXT J @ ASSIGN# 1 TO *
3200 RETURN
3250 IF F$#"A" OR C(6)<2 THEN 3255
3251 A$="*4" @ GOSUB 7000 @ I=B(20) @ IF E(1)=0 THEN
     H1=W(I+5) @ GOTO 3253
3252 H1=ABS(W(I+5)/E(1)) @ IF I=2 THEN H1=H1*T(3) ELSE
     H1=H1*T(4)
3253 A$="*"&VAL$(I) @ GOSUB 7000 @ W(I+5)=H1 @
     A$="*="&VAL$(H1) @ GOSUB 7000
3254 A$="*E8" @ GOSUB 7000
3255 IF Q1=2 THEN RESET 7 @ H1$=D4$[15,20]&D1$&VAL
     $(D2)&"N" @ GOSUB 1520 @ D4$[4,4]="L"
3270 GOSUB 4200 @ GOTO 1380
3300 I3=0 @ S7=0 @ F5$="0" @ ON KEY# 2 GOTO 3390 @ BEEP
     100,10 ! EMTES
```

```
3310 FOR J1=T8/L2+1 TO T9/L2+1 @ H1=(S9-S1(J1))^2 @
     I3=I3+1 @ S7=S7+H1
3320 NEXT J1 @ S6=(S7/I3)^.5 @ IF S6<.00001*(1+K/T(5))
     *B(2) THEN GOTO 3350
3330 IF K=0 THEN GOTO 3350
3340 IF S6>ABS(S9/100*B(2)) THEN DISP "Meas";K;J;
     "Repeated";Z5 @ BEEP @ F5$="1" @ Z5=Z5+1
3345 IF Z5>3*T(8) AND F5$="1" THEN PRINT J;S9;S8;" ??"
     @ F5$="0"
3350 BEEP 200,10 @ OFF KEY# 2 @ RETURN
3390 I6=1 @ GOSUB 4450 @ GOSUB 4550 @ GOTO 3300
3500 T8=INT((B(13)+K/T(5)*(T(6)-B(13)))/L2)*L2
3503 T9=INT((B(14)+K/T(5)*(T(7)-B(14)))/L2)*L2
3505 IF J#1 THEN GOTO 3510 ELSE I=0 @ GOSUB 7900 @ WAIT
     500
3507 IF K<2 OR K=T(5) OR T(5)<7 THEN A$="*H" @ GOSUB
     7000 @ WAIT 3000+B(9)*K/T(5)
3510 MODE 0,1 @ A$="*H" @ GOSUB 7000 @ WAIT
     B(8)+B(9)*K/T(5)
3512 H4=(T(11)-T(10))/2+1 @ H2=(T(2)-T(1))/2+1 @
     H3=(T(13)-T(12))/2+1
3514 ERASE MEMORY -1 ! MESSE
3515 ON ERROR GOTO 3597
3520 MEASURE L1,L2,0,T9+L2
3530 FOR J1=1 TO T9/L2+2
3540 IF NMEAS#J1 THEN GOTO 3540
3550 TO MEMORY J1
3560 IF L1>.2 THEN GOSUB 4400
3570 NEXT J1 @ I6=SPOLL(708) @ IF BIT(I6,0)=1 THEN 2995
3575 ALPHA @ IF L2>.2 THEN 3595
3580 FOR J1=1 TO T9/L2+2
3585 RECALL MEMORY J1
3587 GOSUB 4400 @ NEXT J1
3595 RETURN
```

```
3597 IF ERRN=2 THEN GOSUB 1900 @ PRINT "Ref. repeated"
     @ GOTO 3500 ELSE GOTO 1660
3600 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
3610
3605 ASSIGN# 1 TO D$ @ DISP "File exists" @ ASSIGN# 1
     TO * @ BEEP @ WAIT 2000 @ GOTO 1000
3610 IF ERRN=130 OR ERRN=125 THEN DISP B9$;"not found"
     @ BEEP @ WAIT 2000 @ GOTO 3600
3620 ON ERROR GOTO 1660 @ PLOTTER 1 @ PRINTER 2 @
     C(8)=1 @ B(15)=1 @ C(10)=0
3624 B(6)=T(4)/5 @ GOSUB 4800 @ IF B(4)=1 THEN H1=1 @
     H4=T(4) @ C(7)=1 ELSE GOTO 3634
3626 IF B(6)<H4/50 THEN H1=H1*10 @ H4=H4/10 @ GOTO 3626
3628 IF B(6)<H4/20 THEN H1=H1*2 @ H4=H4/2 @ GOTO 3628
3630 IF H1>4 AND C(8)<B(19) THEN GOSUB 3650 @ GOTO 3624
3632 C(7)=C(7)*T(14)/4*B(6) @ T(14)=6*B(6)
3634 K=T(5) @ GOSUB 8250 @ L(20)=L(20)*T(14)/H4
3636 GOSUB 8000 @ IF D4$[5,5]="0" THEN H1=0 ELSE H1=1
3638 GOSUB 9500
3640 FOR K=0 TO 40 @ T1(K)=0 @ T2(K)=0 @ NEXT K
3645 IF D4$[4,4]="L" THEN GOSUB 1800
3647 RETURN
3650 H2=10/H1 @ A$="*3" @ GOSUB 7000 @ A$="*A"&VAL$(H2)
     @ GOSUB 7000 @ C(8)=C(8)+1
3660 A$="*5" @ GOSUB 7000 @ A$="*I" @ GOSUB 7000 @ DISP
     "Please Wash Tip" @ WAIT 5000
3665 A$="*4" @ GOSUB 7000 @ A$="*D" @ WAIT 5000 @ GOSUB
     7000 @ A$="*E 6" @ GOSUB 7000
3670 T(4)=T(4)/H1 @ T(14)=T(14)/H1 @ C(7)=H1
3680 RETURN
3700 T1(41)=0 @ T2(41)=0 @ H4=W1(3)*T(4)/B(7)
3710 FOR J=1 TO T(0) @ T1(41)=T1(41)+S4(J) @
     T2(41)=T2(41)+S5(J) @ NEXT J
```

```
3720 T1(41)=T1(41)/T(0) @ T2(41)=T2(41)/T(0) @
     R9=R9/T(0) @ R9=R9^.5 @ R8=(R8/T(0))^.5
3730 DISP USING "DD,X,5D.DD,XXX,DDD.DD,XXX,.6D" ;
     K,H4,B(3),R8 @ RETURN
3800 S9=0 @ S8=0 @ F5$="0" ! CH
3810 FOR J=1 TO T(0)
3820 S9=S9+(T1(41)-S4(J))^2 @ S8=S8+(T2(41)-S5(J))^2 @
     NEXT J @ S9=(S9/T(0))^.5
3830 IF S9<ABS(.00005*B(1)) OR Q1#1 THEN GOTO 3860
3850 IF S9>ABS(T1(41)/100*B(1)) THEN DISP "Average";K;"
     repeated";Z5 @ BEEP @ F5$="1"
3855 Z5=Z5+T(0) @ IF Z5>2*T(8) AND F5$="1" THEN 3870
3860 RETURN
3870 PRINT "Average nonreliable" @ FOR J=1 TO T(0) @
     PRINT "Msg";K;J;" ";S4(J);S5(J) @ NEXT J
3875 F5$="0" @ GOTO 3860
4200 IF D4$[5,5]="0" THEN 4205
4201 PRINTER 2 @ PRINT @ PRINT @ PRINT TAB(7);E4$ @
     GRAPH @ COPY
4202 H1=0 @ GOSUB 9500 @ PRINT T1$ @ ON ERROR GOTO 1660
4205 PRINT @ PRINT @ PRINT "No.  Range 1   Range 2
     [Substr]"
4210 PRINT "     TN(1/sec) TN(1/sec)    M" @ PRINT
4220 K=T(5) @ GOSUB 8250
4230 A$="PFTN.KSYS"
4250 GOSUB 8500
4260 LORG 0 @ CSIZE 3 @ GOSUB 5500
4270 FOR K=0 TO T(5) @ GOSUB 8250 @ IF K=0 THEN
     T1(40)=H3
4290 MOVE H4,H3 @ LABEL "*" @ MOVE H4,J1 @ LABEL "+"
4310 PRINTER IS 2 @ PRINT USING "DD,X,S5D.D,X,
     S5D.D,XX,5D.DD" ; K,H3,J1,H4
4312 PRINT# 1 ; H3,J1,H4
4320 NEXT K @ PRINTER IS 2
```

```
4330 PRINT @ IF D4$[5,5]="1" THEN PRINT @ PRINT
     TAB(7);E4$ @ COPY @ PRINT "Faktor:";C(7) @ PRINT
4350 ASSIGN# 1 TO *
4360 GOSUB 6000 @ PRINT @ PRINT @ PRINT @ RETURN
4400 I3=0 @ H1=0
4410 FOR I2=T(1) TO T(2) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4415 S1(J1)=H1/H2 @ H1=0
4420 FOR I2=T(12) TO T(13) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4425 S2(J1)=H1/H3 @ H1=0
4430 FOR I2=T(10) TO T(11) STEP 2 @ H1=H1+VALUE(I2) @
     NEXT I2
4435 S1(J1)=S1(J1)-H1/H4 @ S2(J1)=S2(J1)-H1/H4
4440 RETURN
4450 DISP "1=Exit 2=New Stdev 3=Rep.Single" @ DISP
     "4=Rep.Aver. 5=End Me  6=go on"
4455 DISP "7=Test Msg 8=Next Ready";
4460 INPUT H1@ RETURN
4500 IF F$="A" THEN GOTO 2250
4510 GOSUB 4450 @ ON ERROR GOTO 1660 @ I6=2
4520 F6$="0" @ IF H1#3 THEN 4550
4530 F6$="1" @ DISP "No. of Single Measurement" @ INPUT
     J@ FOR J1=1 TO 80 @ S3(J1)=0 @ NEXT J1
4540 R9=R9^2*T(0)/(1.2+T(0)) @ R8=R8^2*T(0)/(1.2+T(0))
     @ GOTO 2310
4550 IF H1=5 THEN 2995
4555 IF H1=7 THEN 2000
4560 IF H1=2 THEN DISP "Max. Nonlin./Noise 2 Inputs";@
     INPUT B(1),B(2)@ GOTO 4590
4565 IF H1=8 THEN GOSUB 900 @ GOTO 4590
4570 IF H1=3 AND I6=1 THEN F6$="2" @ GOTO 4590
4580 IF H1=4 THEN 2250
4590 RETURN
```

```
4600 S9=0 @ S8=0 @ I1=0
4620 H1=S1(1) @ H2=S2(1)
4630 FOR J1=2 TO T9/L2+1
4635 H3=S1(J1) @ H4=S2(J1)
4640 S1(J1)=(S1(J1+1)-H1)/(2*L2) @ S2(J1)=(S2(J1+1)-
     H2)/(2*L2) @ IF J1>T9/L2+1 THEN GOTO 4710
4700 IF J1>=T8/L2+1 THEN I1=I1+1 @ S9=S9+S1(J1) @
     S8=S8+S2(J1)
4710 H1=H3 @ H2=H4 @ NEXT J1 @ I3=0 @ S8=S8/I1 @
     S9=S9/I1
4730 DISP USING "DD,X,DD,XX,SZ.6D,XXX,SZ.6D" ;
     K,J,S9,S8
4740 S4(J)=S9 @ S5(J)=S8
4750 RETURN
4800 F6$="0" @ A5=0 @ A6=0 @ MODE 0,1
4810 LAMBDA L(1),L(2),L(3),L(4),L(5),L(6),L(7),
     L(8),L(9),L(10)
4820 TIME SCALE 0 TO T(7)
4830 ABSORBANCE
4832 IF B(3)>T(3)/3 THEN B(3)=T(3)/3
4833 IF B(6)>T(4)/3 THEN B(6)=T(4)/3
4835 W1(3)=B(7)*B(6)/T(4)
4836 W1(2)=B(7)*B(3)/T(3)
4840 IF W1(3)<=0 THEN GOTO 1900
4845 IF C(9)#1 THEN 1000
4850 I=0 @ GOSUB 7980
4860 A$="*H" @ GOSUB 7000
4865 WAIT 5000
4870 J=0 @ K=INT(T(5)/2) @ T9=T(7) @ T8=B(13) @ GOSUB
     3510
4920 H1=0 @ H2=0
4930 FOR J1=1 TO 4 @ H1=S1(J1)+H1 @ H2=S2(J1)+H2 @ NEXT
     J1 @ H1=H1/4 @ H2=H2/4
```

```
5050 IF D4$[5,5]#"0" AND A5#1 THEN A$="PFOD.KSYS" @
     GOSUB 8500 @ A5=1
5055 DISP "Initial OD Range 1 and Range 2"
5056 DISP USING "6X,SD.5D,6X,SD.5D" ; H1,H2 @
     H1=H1/T(15)*1000 @ H2=H2/T(16)*1000
5057 IF ABS(H1)>99999 OR ABS(H2)>99999 THEN DISP
     "[E]";H1;H2;"uM" @ GOTO 5060
5058 DISP USING "3A,X,S5D.3D,4X,S5D.3D,2A" ; "[E]",H1,
     H2,"uM" @ DISP "Deriv. Range1 and Range 2"
5060 IF D4$[5,5]#"0" THEN F6=1 @ GOSUB 6500 @ F6=2 @
     GOSUB 6500
5150 K=0 @ J=0 @ GOSUB 4600 @ GOSUB 3300 @
     H4=ABS(S6/S9*100) @ IF H4>999 THEN H4=999
5160 DISP USING "14A,X,.6D,3A,3D.D,A" ; "Noise  Range
     1",ABS(S6)," = ",H4,"%" @ H1=0 @ H2=0
5170 FOR J=2 TO 6 @ H1=H1+S1(J) @ NEXT J @ H1=H1/5 @
     A1=H4
5180 FOR J=T(7)/L2-3 TO T(7)/L2+1 @ H2=H2+S1(J) @ NEXT
     J @ H2=H2/5 @ H3=INT(ABS((H1-H2)/H1*100))
5190 DISP "Nonlinearity  : ";H3;"%" @ A2=H3 @ DISP @
     DISP
5300 J1=0 @ IF D4$[5,5]#"0" THEN GOSUB 5400
5310 IF J1=9 THEN 4810
5350 IF A5#2 THEN A$="PFAB.KSYS" @ GOSUB 8500 @ A5=2
5360 F6=1 @ GOSUB 6500 @ F6=2 @ GOSUB 6500
5390 GOSUB 5400 @ IF J1=9 THEN 4810
5395 RETURN
5400 OFF KEY# 8 @ ON KEY# 2,"Go on" GOTO 5490 @ ALPHA
5402 ON KEY# 5,"[E]" GOTO 5440 @ ON KEY# 6,"Activ" GOTO
     5445 @ ON KEY# 7,"[I]" GOTO 5435
5405 ON KEY# 3,"Graph" GOTO 5420 @ ON KEY# 4,"Alpha"
     GOTO 5430 @ KEY LABEL
5407 IF F$="A" AND C(10)=0 THEN J1=1 ELSE J1=0
```

```
5410 IF J1=0 THEN 5410 ELSE BEEP 300,20 @ WAIT 3000 @
     GOTO 5450
5420 GRAPH @ J1=0 @ GOTO 5410
5430 ALPHA @ J1=0 @ GOTO 5410
5435 DISP B(6);"new";@ INPUT B(6)@ J1=9 @ GOTO 5490
5440 DISP B(3);"new";@ INPUT B(3)@ B(5)=B(3) @ J1=9 @
     GOTO 5490
5445 DISP B(10);"new";@ INPUT B(10)@ GOTO 5400
5450 IF B(4)=0 OR Q1=2 THEN 5490
5454 IF B(6)<T(4)/2000 THEN B(6)=T(4)/2000 @ GOTO 5490
5455 IF ABS(S9)<B(10)/2 THEN H1=.25 @ GOTO 5492
5465 IF ABS(S9)>1.3*B(10) THEN H1=1.3 @ GOTO 5492
5470 IF ABS(S9)<.75*B(10) THEN H1=.75 @ GOTO 5492
5490 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ RETURN
5492 B(6)=B(6)*H1 @ A6=A6+1 @ CLEAR @ DISP
     "Inhib.conc.:";B(6) @ IF A6>6 THEN J1=0 ELSE J1=9
5493 GOTO 5490
5495 GOSUB 7250 @ GOTO 5400
5500 D$=D4$[15,20]&D1$&VAL$(D2)&"."&B9$ @ ON ERROR GOTO
     1660
5550 CREATE D$,3+INT(T(5)/10+.5),256
5560 ASSIGN# 1 TO D$
5570 PRINT# 1 ; D1$,D2,D3$,D4$,E1$,E2$,T1$,T( ),L( ),B( )
5660 RETURN
6000 A1=0 @ A2=0 @ A3=0 @ A4=0 @ A5=0 @ A6=0
6050 FOR K=1 TO T(5) @ GOSUB 8250 @ H3=T1(40)-H3
6090 IF H3<=0 OR H4=0 THEN GOTO 6200 ELSE H1=1/H4 @
     H2=T1(40)/H3
6120 A1=A1+1 @ A2=A2+H1 @ A3=A3+H2 @ A4=A4+H1*H2 @
     A5=A5+H1*H1 @ A6=A6+H2*H2
6200 NEXT K
6210 IF A1=0 THEN GOTO 6400
6220 H1=A5/A1-(A2/A1)^2
```

```
6230 H2=A6/A1-(A3/A1)^2
6240 H3=(A4/A1-A2/A1*A3/A1)/H1
6250 H4=A3/A1-H3*A2/A1
6260 H5=H3*H1^.5/H2^.5
6280 H1=ABS(1/(H4/H3)) @ H2=ABS(1/H4)
6290 IF ABS(H1)>9999 OR ABS(H2)>99999 OR ABS(H5)>1 THEN
     GOTO 6400
6300 PRINT USING "15A,7X,3D.2D,X,3A" ; "Inhibition
     =",(1-H2)*100," % "
6310 PRINT USING "17A,2X,5D.3D,X,3A" ; "Inhibition
     Const=",H1," M"
6320 PRINT USING "15A,9X,3D.4D" ; "Correlation   =",H5
6400 PRINT @ PRINT @ RETURN
6500 PLOT 0,0,2
6520 FOR J1=2 TO T9/L2+1
6530 H4=(J1-1)*L2
6540 ON F6 GOTO 6550,6560,6570
6550 H3=S1(J1) @ GOTO 6580
6560 H3=S2(J1) @ GOTO 6580
6570 H3=S3(J1)
6580 PLOT H4,H3,1
6590 NEXT J1 @ PENUP @ PLOT 0,0,2 @ RETURN
7000 OFF KEY# 2 @ OFF KEY# 3 @ OFF KEY# 4 @ OFF KEY# 5
     @ OFF KEY# 6 @ OFF KEY# 7 @ OFF KEY# 8
7005 SET TIMEOUT 7;3000
7010 ON TIMEOUT 7 GOTO 7090
7015 A1$=A$[2,2]
7020 I5=SPOLL(708)
7025 IF BIT(I5,7) THEN WAIT 500 @ GOTO 7020
7030 IF BIT(I5,5) THEN DISP "Empty" @ F5$="2" @ C(15)=0
     @ GOTO 7082
7032 IF BIT(I5,4) THEN DISP "Full" @ C(15)=0
7035 IF BIT(I5,3) THEN C(15)=0
7040 A$=A$&"," @ OUTPUT 708 ;A$
```

```
7052 IF A1$="1" THEN M=1
7054 IF A1$="2" THEN M=2
7060 IF A1$="3" THEN M=3
7061 IF A1$="B" OR A1$="D" THEN C(M)=C(M)-W(M+10)
7062 IF A1$="C" THEN C(M)=C(M)+W(M+10)
7063 IF A1$="F" THEN C(M)=0
7064 IF A1$="G" THEN C(M)=W(M)
7065 IF A1$="H" THEN C(1)=C(1)-W1(1) @ C(2)=C(2)-W1(2)
     @ C(3)=C(3)-W1(3)
7070 OFF TIMEOUT 7
7080 RETURN
7082 IF A1$="F" OR A1$="B" OR A1$="D" THEN 7080 ELSE
7040
7090 DISP "Switch on ASSAYOMATE" @ BEEP @ WAIT 5000 @
     RESET 7 @ GOTO 7000
7100 DISP A$ @ A$="*TF84A " @ I6=W1(4) @ GOSUB 7200
7110 I6=INT(W1(4)/(W1(1)*48000/W(1)))+257 @ GOSUB 7200
7115 IF W1(2)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(2)*48000/W(2)))+257
7120 GOSUB 7200 @ IF W1(3)=0 THEN I6=60258 ELSE
     I6=INT(W1(4)/(W1(3)*48000/W(3)))+257
7125 GOSUB 7200
7130 GOSUB 7000
7190 RETURN
7200 B$="    " @ H1$="0123456789ABCDEF"
7205 FOR J1=4 TO 1 STEP -1
7210 I=I6 MOD 16 @ B$[J1,J1]=H1$[I+1,I+1] @ I6=I6-I @
     I6=I6/16 @ NEXT J1
7215 A$=A$&"="&B$[3,4]&"="&B$[1,2]
7220 RETURN
7250 M=0 @ A$="*WFCC8" @ GOSUB 7000 @ GOSUB 7700 @ GOTO
     7480
7280 ON KEY# 1,"Give" GOTO 7430 @ ON KEY# 2,"Suck" GOTO
     7420
```

```
7300 ON KEY# 3,"Back" GOTO 7400 @ ON KEY# 4,"Empty"
     GOTO 7460
7320 ON KEY# 5,VAL$(W(M+10)) GOTO 7360
7330 ON KEY# 6,VAL$(C(M)) GOTO 7440
7340 ON KEY# 7,"Retur" GOTO 7355
7350 ON KEY# 8,"Syr."&VAL$(M) GOTO 7480
7352 KEY LABEL @ GOTO 7280
7355 OFF KEY# 1 @ RETURN
7360 CLEAR @ DISP "Volume per Key";@ INPUT H1$@
     W(M+10)=VAL(H1$) @ A$="*A"&H1$ @ GOSUB 7000
7370 GOTO 7280
7400 A$="*D" @ GOSUB 7000 @ GOTO 7280
7420 A$="*C" @ GOSUB 7000 @ GOTO 7280
7430 A$="*B" @ GOSUB 7000 @ GOTO 7280
7440 A$="*G" @ GOSUB 7000 @ GOTO 7280
7460 A$="*F" @ GOSUB 7000 @ GOTO 7280
7480 M=M+1 @ IF M>3 THEN M=1
7485 A$="*"&VAL$(M) @ GOSUB 7000 @ GOTO 7280
7500 CLEAR @ DISP "No. of Cycles and Syringe (1-3)";@
     INPUT H1,M
7510 IF M<1 OR M>3 THEN 7500
7520 A$="*"&VAL$(M) @ GOSUB 7000
7530 A$="*E"&VAL$(H1) @ GOSUB 7000
7535 IF H1 MOD 2#0 THEN C(M)=0 ELSE C(M)=W(M+5)
7540 RETURN
7700 SET TIMEOUT 7;2000 @ A$=""
7710 ON TIMEOUT 7 GOTO 7790
7720 ENTER 708 USING "#,B" ; H1@ A$=A$&CHR$(H1) @ IF
     H1=13 OR LEN(A$)>40 THEN 7740
7730 GOTO 7720
7740 OFF TIMEOUT 7 @ ON ERROR GOTO 7790
7750 C(1)=VAL(A$[10,16])
7760 C(2)=VAL(A$[20,26]) @ C(3)=VAL(A$[30,36]) @
     C(15)=1
```

```
7790 ON ERROR GOTO 1600 @ RETURN
7900 IF K=0 THEN N3=0 @ GOTO 7970
7910 ON T(19) GOTO 7920,7930,7950,7940,7960
7920 N3=4*L3^(K-1) @ GOTO 7970
7930 N3=K*300/T(5) @ IF K>T(5)/2 THEN N3=150+INT(K-
     T(5)/2)*600/T(5) @ GOTO 7970 ELSE GOTO 7970
7940 K1=T(5)+1-K @ N3=1/K1*600 @ GOTO 7970
7950 N3=5+K*100/T(5)+3*L3^(K-1) @ GOTO 7970
7960 N3=N3/L(20)
7970 W1(3)=N3*L(20)/10000
7975 W1(2)=B(7)*B(3)/T(3)*(1-B(17)*(T(5)-K)/T(5))
7980 W1(1)=B(7)-W1(2)-W1(3) @ IF I=9 THEN RETURN
7985 A$="*K"&VAL$(W1(1))&","&VAL$(W1(2))&",
     "&VAL$(W1(3)) @ GOSUB 7100
7990 GOSUB 7000 @ RETURN
8000 S1(1)=4 @ S1(2)=.3 @ S1(3)=.25 @ I=9 @ A$="*WFCC8"
     @ GOSUB 7000 @ GOSUB 7700
8010 FOR K=0 TO T(5) @ GOSUB 7900 @ FOR M=1 TO 3 @
     S1(M)=S1(M)+W1(M)*T(8)*1.1 @ NEXT M @ NEXT K
8060 FOR M=1 TO 3 @ IF S1(M)>C(M) THEN DISP "Syringe
      ";M;" is filled" @ GOSUB 8100
8070 NEXT M @ RETURN
8100 K=B(20) @ A$="*"&VAL$(M) @ GOSUB 7000 @ IF M#K
     THEN 8120
8105 S1(K)=S1(K)+.2 @ IF S1(K)>W(K) THEN S1(K)=W(K)
8107 IF S1(K)>B(19) THEN S1(K)=B(19)
8110 A$="*="&VAL$(S1(K)) @ GOSUB 7000 @ W(K+5)=S1(K)
8120 A$="*G" @ GOSUB 7000
8190 RETURN
8200 I=0 ! BER. [E] [S]
8205 GOSUB 7900
8210 H2=T(3)*W1(2)/B(7)
8220 H4=T(4)*W1(3)/B(7) ! [S]
```

```
8230 H3=T1(K)*1000000/T(17)/H2 @ J1=T2(K)*1000000/
     T(18)/H2
8240 RETURN
8250 I=9 @ GOSUB 8205
8260 RETURN
8500 RESET 7
8510 ASSIGN# 1 TO A$ @ READ# 1 ; H3,H4,X0,X1,X2,X3,Y0,
     Y1,Y2,Y3,E3$,E4$
8550 GRAPH @ GCLEAR @ SCALE 0,100,0,100 @ IF X2=0 THEN
     X2=(X1-X0)/5
8560 MOVE 30,0 @ LDIR 0 @ CSIZE 3 @ LABEL E3$ @ F5=(X1-
     X0)/7 @ F6=(Y1-Y0)/10
8590 SCALE X0-F5,X1,Y0-F6,Y1 @ XAXIS Y1,X3,X0,X0+X3 @
     XAXIS Y1,X2,X0+X3,X1
8620 XAXIS Y0,X3,X0,X0+X3 @ XAXIS Y0,X2,X0+X3,X1 @
     YAXIS X1,Y3,Y0,Y0+Y3
8650 YAXIS X1,Y2,Y0+Y3,Y1 @ YAXIS X0,Y3,Y0,Y0+Y3 @
     YAXIS X0,Y2,Y0+Y3,Y1
8660 FOR K=X0+X3 TO X1 STEP X2
8670 MOVE K-F5/5,Y0-(Y1-Y0)/20
8680 LABEL VAL$(K) @ NEXT K
8690 FOR K=Y0+Y3 TO Y1 STEP Y2
8700 MOVE X0-(X1-X0)/10,K-F6/2
8710 LABEL VAL$(K) @ NEXT K
8900 ASSIGN# 1 TO * @ RETURN
9500 PRINT @ PRINT @ PRINT
9510 PRINT "Measurement ";D4$[15,20];"  ";D1$;D2 @
     PRINT "Diskette:";B9$;"  ";D3$
9540 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9542 IF H1=0 THEN 9700
9545 PRINT "Time Window";TAB(16);T(6);
     "bis";TAB(24);T(7);"sec"
9548 PRINT "Max.Stdev";B(1);" / Nonlin.";B(2);"%"
9550 PRINT "Range 1 from   ";T(1);" to";T(2);"nm"
```

```
9555 PRINT "Range 2 from   ";T(12);" to";T(13);"nm"
9560 PRINT "[";E1$;"]";TAB(23);T(3);"nM"
9565 PRINT "in the Assay :  ";B(17);TAB(23);B(3);"nM"
9580 PRINT "[";E2$;"]";TAB(23);T(4);"M"
9585 PRINT "Max. Conc. in Assay:  ";T(14);"M"
9657 PRINT "Assay Volume :";TAB(25);B(7);"ml"
9690 FOR I=1 TO 32 @ PRINT "-";@ NEXT I @ PRINT
9700 PRINTER IS 2 @ RETURN
```

What is claimed is:

1. A device for characterizing chemical reactions, comprising:

supply means for supplying an accurately metered plurality of solutions containing chemical reactants which when combined have a defined kinetic constant associated with their reaction;

mixing chamber means connected to said supply means for mixing said plurality of solutions from said supply means;

temperature control means associated with said mixing chamber means for controlling temperature of said mixing chamber means;

reaction chamber means where a substantial portion of the reaction between said reactants occurs, said reaction chamber means connected to said mixing chamber;

detection means for measuring a physical parameter that is a function of concentration of at least one of said reactants and reaction products in said reaction chamber means;

computer means coupled to said supply means, to said mixing chamber means, to said temperature control means, and to said detection means, for automatically controlling the supply means to provide a plurality of sets of simultaneous metered volumes of said solutions to said mixing chamber means, each set corresponding to a predefined ratio of said reactants, for automatically controlling the temperature of said mixing chamber means, and for automatically causing said detection means to make measurements at a plurality of times for each set of said simultaneous metered volumes to determine changes of said concentration in time for each set;

said computer means comprising processing means for determining said kinetic constant.

2. A device as in claim 1 wherein said supply means comprises a plurality of calibrated syringe means and a plurality of containers for said solutions, said plurality of syringe means for sucking solutions from said containers and for metering out preselected volumes of said solutions sucked in from said containers;

said supply means further comprising valve means connected to each of said syringe means and said mixing chamber means, for receiving said solutions from said syringe means, for combining said solutions in predetermined volumetric ratios established by the rate at which said plurality of syringe means meter out said solutions, and for passing said combined solutions to said mixing chamber means;

each of said syringe means having independent drive means for controlling the rate at which each said syringe meters out said solutions;

said drive means being coupled to said computer means and controlled thereby.

3. A device as in claim 2 wherein each of said drive means comprises a dedicated processor for controlling the rate at which each syringe meters out said solutions.

4. A device as in claim 1 wherein said mixing chamber means comprises:

a mixing chamber having a plurality of inlet ports and at least one exit port, said mixing chamber for containing said solutions during mixing;

mixing head means, at least a portion of which comprises a magnetic material, said mixing head means located inside said mixing chamber means, said mixing head means for mixing said solutions when said mixing head means is rotated inside said mixing chamber;

mixing head driver means located outside said mixing chamber, for creating a moving magnetic field inside said mixing chamber;

said mixing head means and said mixing head driver means arranged such that said mixing head means is rotated by magnetic induction due to said moving magnetic field.

5. A device as in claim 4 wherein said mixing head means has a rotation axis, and wherein said mixing chamber and said mixing head means define therebetween an annular, slot-like space forming a shear mixing zone coaxial to said rotation axis of said mixing head means, with said mixing head means being floatingly arranged within said mixing chamber.

6. A device as in claim 4 further comprising housing means for containing said mixing chamber, said housing means configured for causing thermal communication between a thermostating fluid and said mixing chamber in order to control the temperature of said mixing chamber.

7. A device as in claim 6 further comprising thermostating means for controlling the temperature of said thermostating fluid.

8. A device as in claim 4 further comprising a supply tube, said supply tube having one end connected to one of said inlet ports, said supply tube immediately before its one end connected to said inlet port having an elastic orifice which is expanded by increasing fluid pressure in said supply tube.

9. A device for determining kinetic constants of chemical reactions, comprising:

mixing means for mixing at least two chemical reactants in a succession of controlled quantities, said reactants forming a reaction product;

measurement means for acquiring data regarding changes in concentration of at least one of said reactants and said reaction product after said mixing; and

processor means having an equation interpreter for analyzing kinetic equations input by a user of said device, said processor means for calculating said kinetic constants based on said kinetic equations and said data from said measurement means.

10. A device for determining kinetic constants of chemical reactions, comprising:

mixing means for mixing two chemical reactants in a succession of controlled quantities, said reactants forming a reaction product;

measurement means for acquiring data regarding changes in concentration of at least one of said reactants and said reaction product after said mixing; and

processor means having a plurality of built-in kinetic models for chemical reactions, said processor means comprising input means for user selection of at least one of said kinetic models, said processor means for calculating said kinetic constants based on said kinetic model selected by said user and said data from said measurement means.

11. A device for characterizing chemical reactions, comprising:

supply means for supplying an accurately metered plurality of solutions containing chemical reactants which when combined have a defined kinetic constant associated with their reaction;

mixing means connected to said supply means for mixing said plurality of solutions from said supply means;

reaction chamber means where a substantial portion of the reaction between said reactants occurs, said reaction chamber means connected to said mixing means;

detection means for measuring a physical parameter that is a function of concentration of at least one of said reactants and reaction products in said reaction chamber means;

computer means coupled to said supply means, to said mixing means, and to said detection means, for automatically controlling the supply means to provide a plurality of sets of simultaneous metered volumes of said solutions to said mixing chamber means, each set corresponding to a predefined ratio of said reactants, for automatically controlling the temperature of said mixing chamber means, and for automatically causing said detection means to make measurements at a plurality of times for each set of said simultaneous metered volumes to determine changes of said concentration in time for each set;

said computer means comprising processing means for determining said kinetic constant; and

said computer means operating according to an instruction set that after an initial measurment by said measurement means, said computer means determines if the measurement corresponds to a useful concentration of said reactants, and if a concentration is too large for a particular reactant, dilutes that reactant until that reactant has a concentration in a useful range.

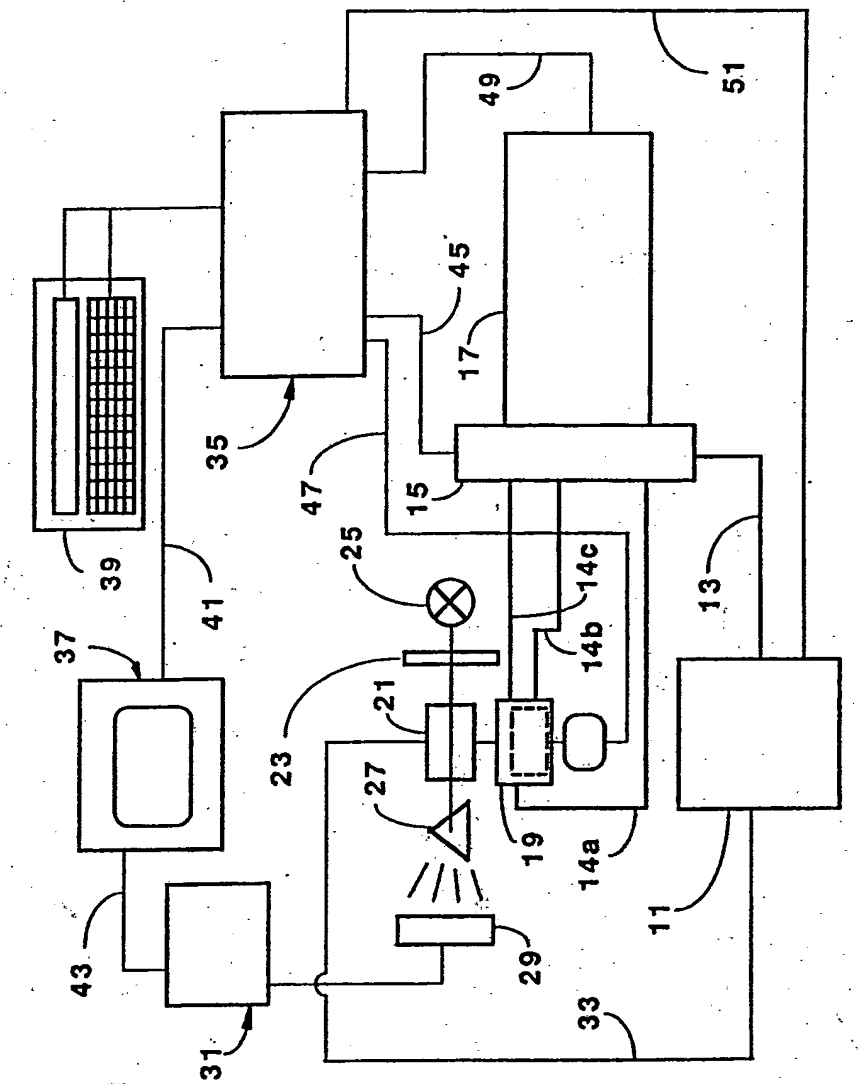

51
49
45
17
35
47
15
13
25
14c
14b
39
41
37
23
21
27
19
14a
11
43
29
33
31

Fig.1

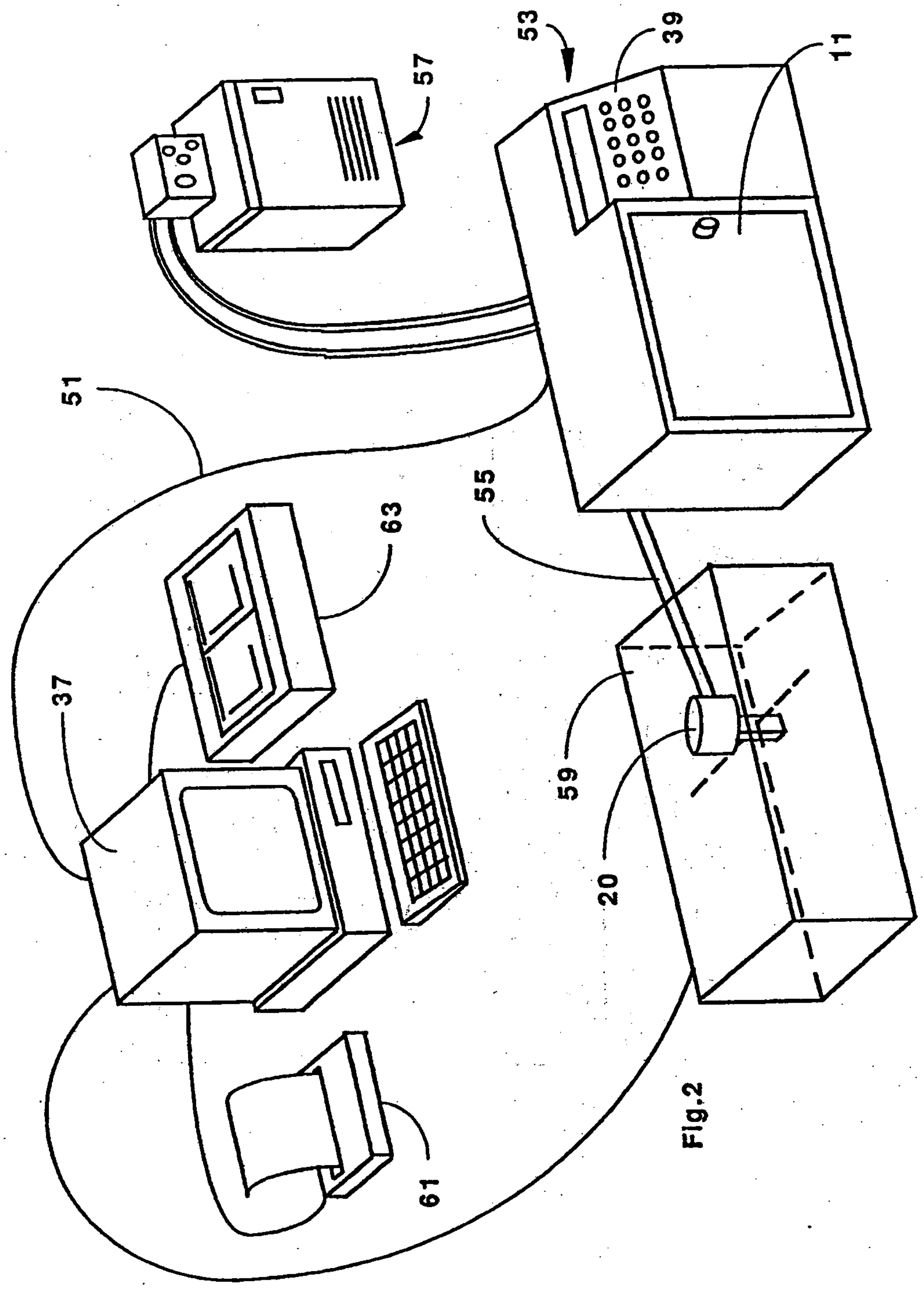


Fig.2

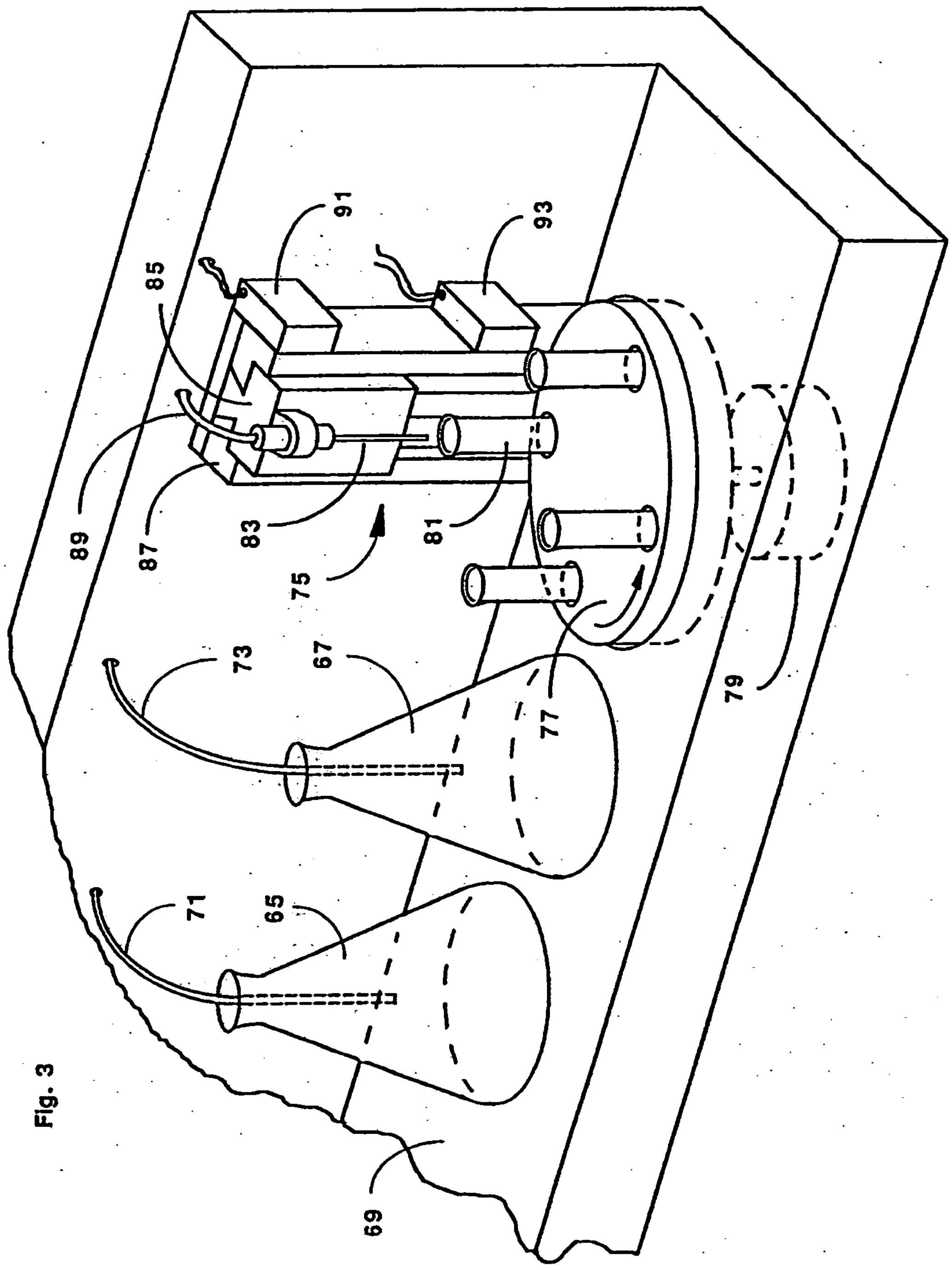
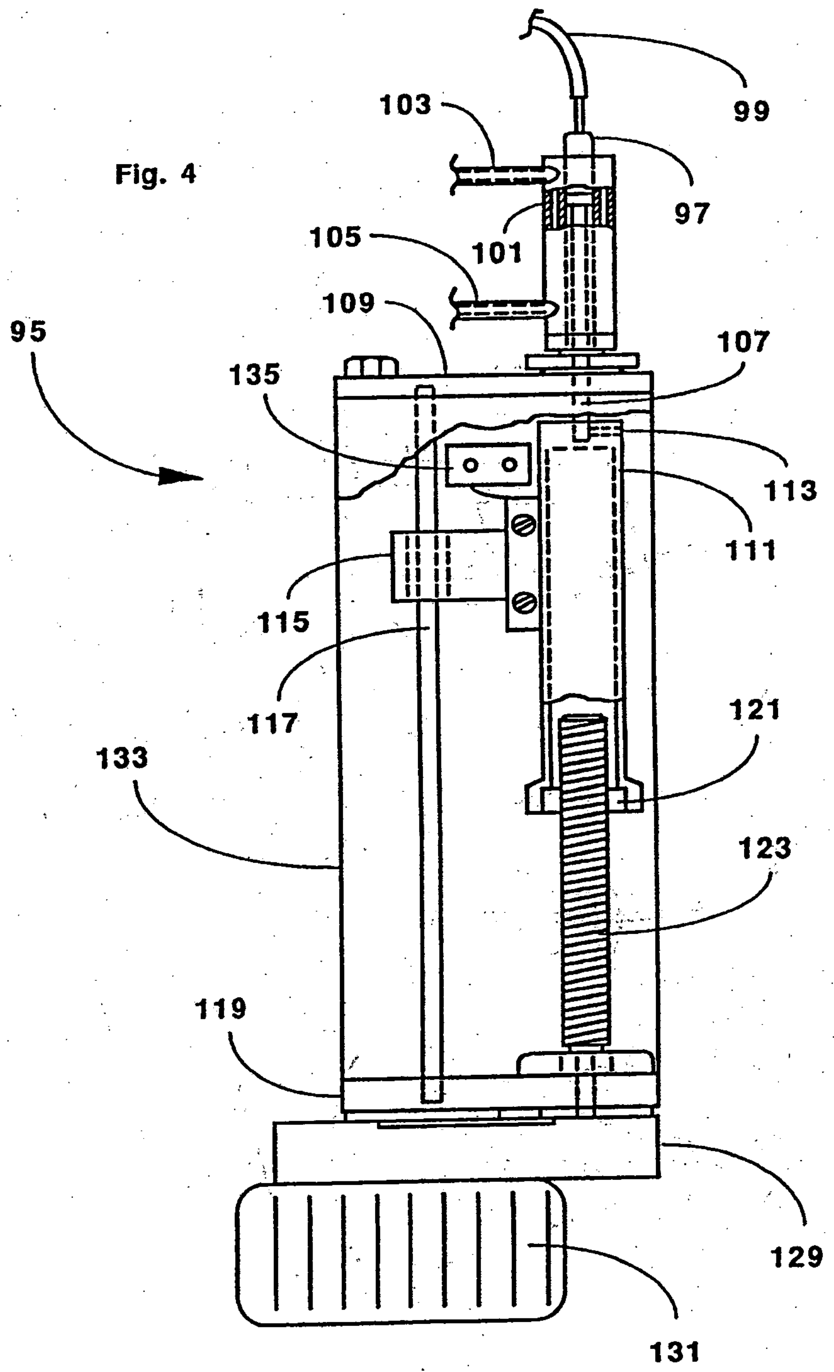


Fig. 3

Fig. 4


99
103
97
101
105
109
95
107
135
113
111
115
117
121
133
123
119
129
131

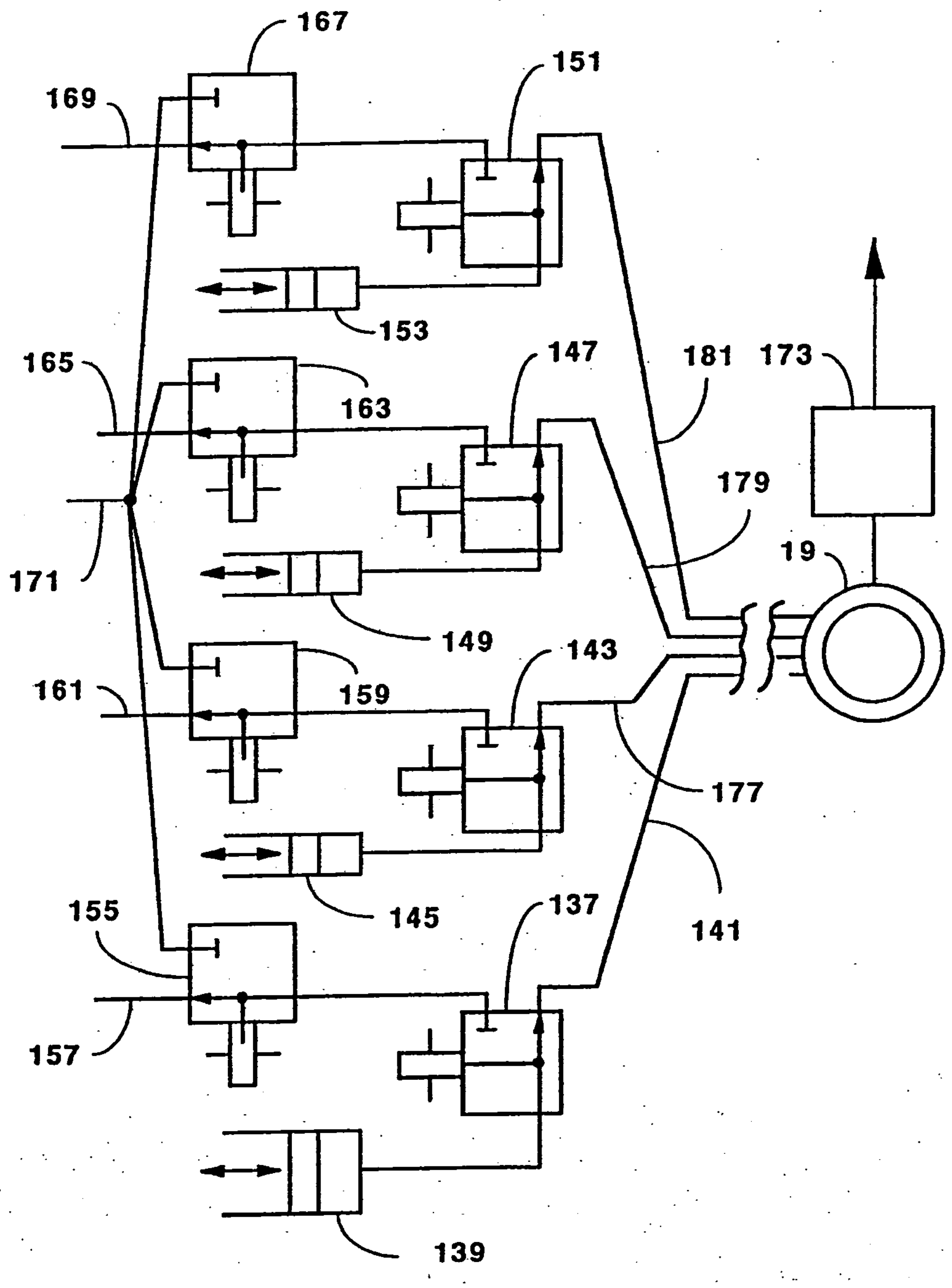

167
151
169
153
165
163
147
181
173
179
171
149
19
143
161
159
177
145
141
155
137
157
139

Fig. 5

Fig. 6

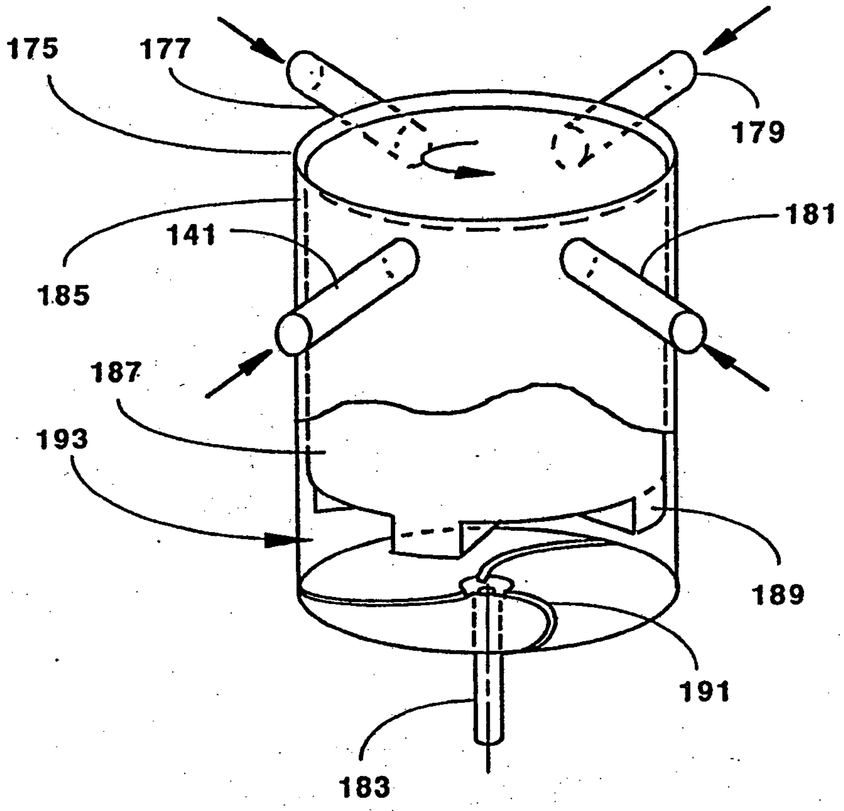


Fig. 7 (A)

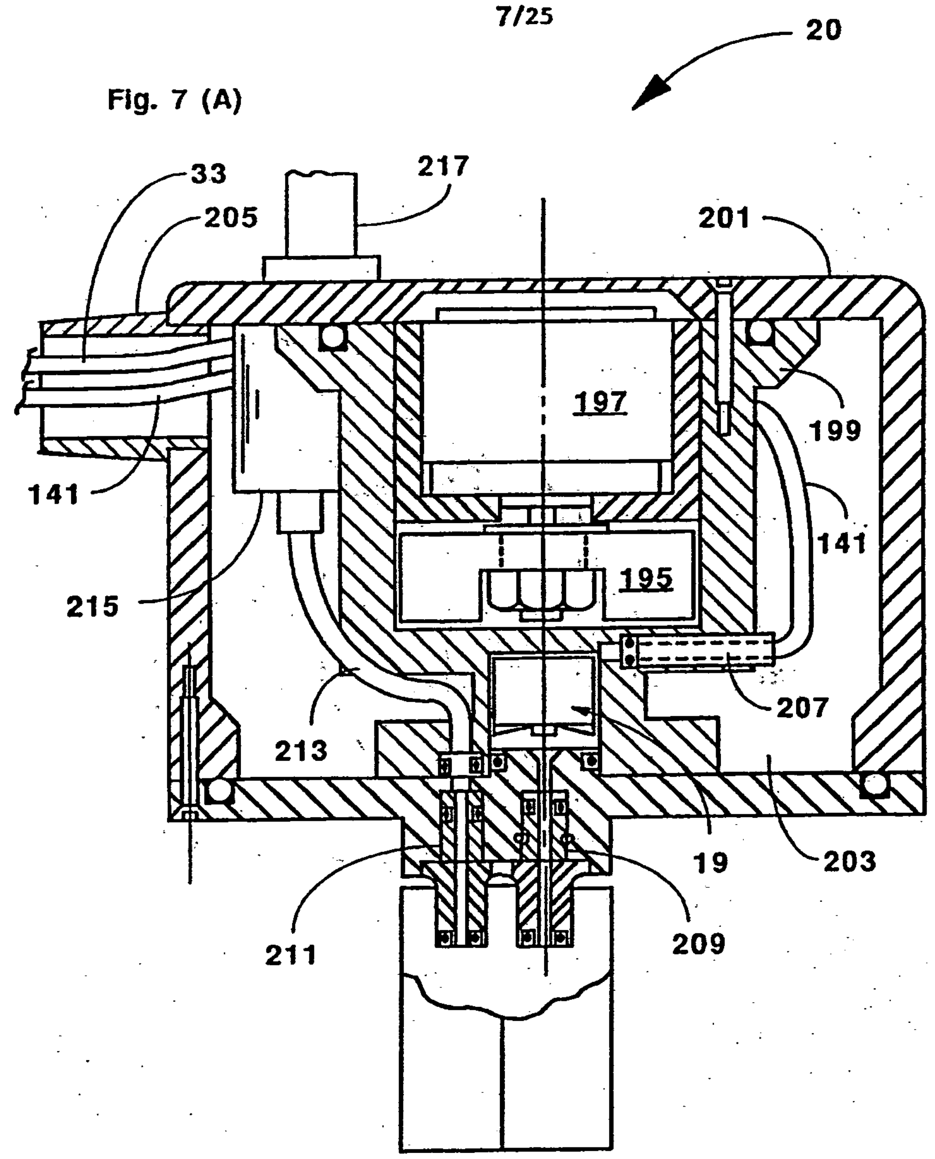
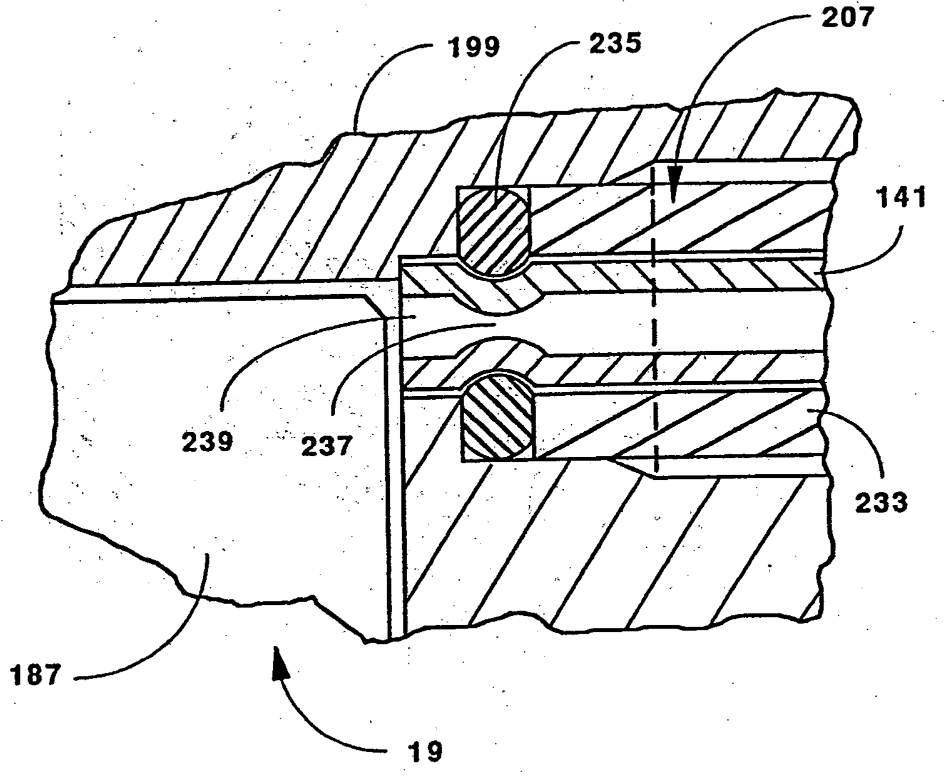


Fig. 7 (B)



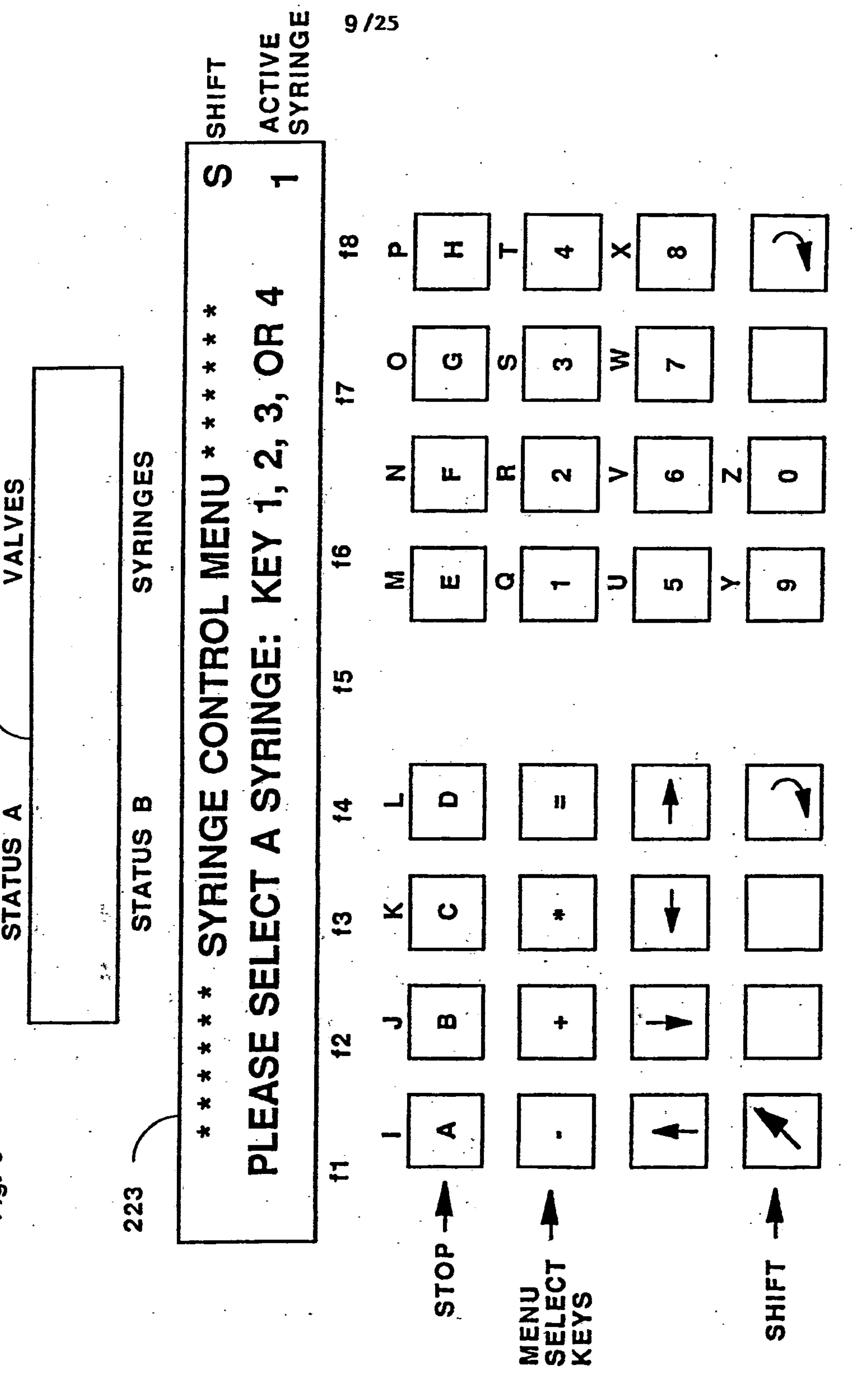
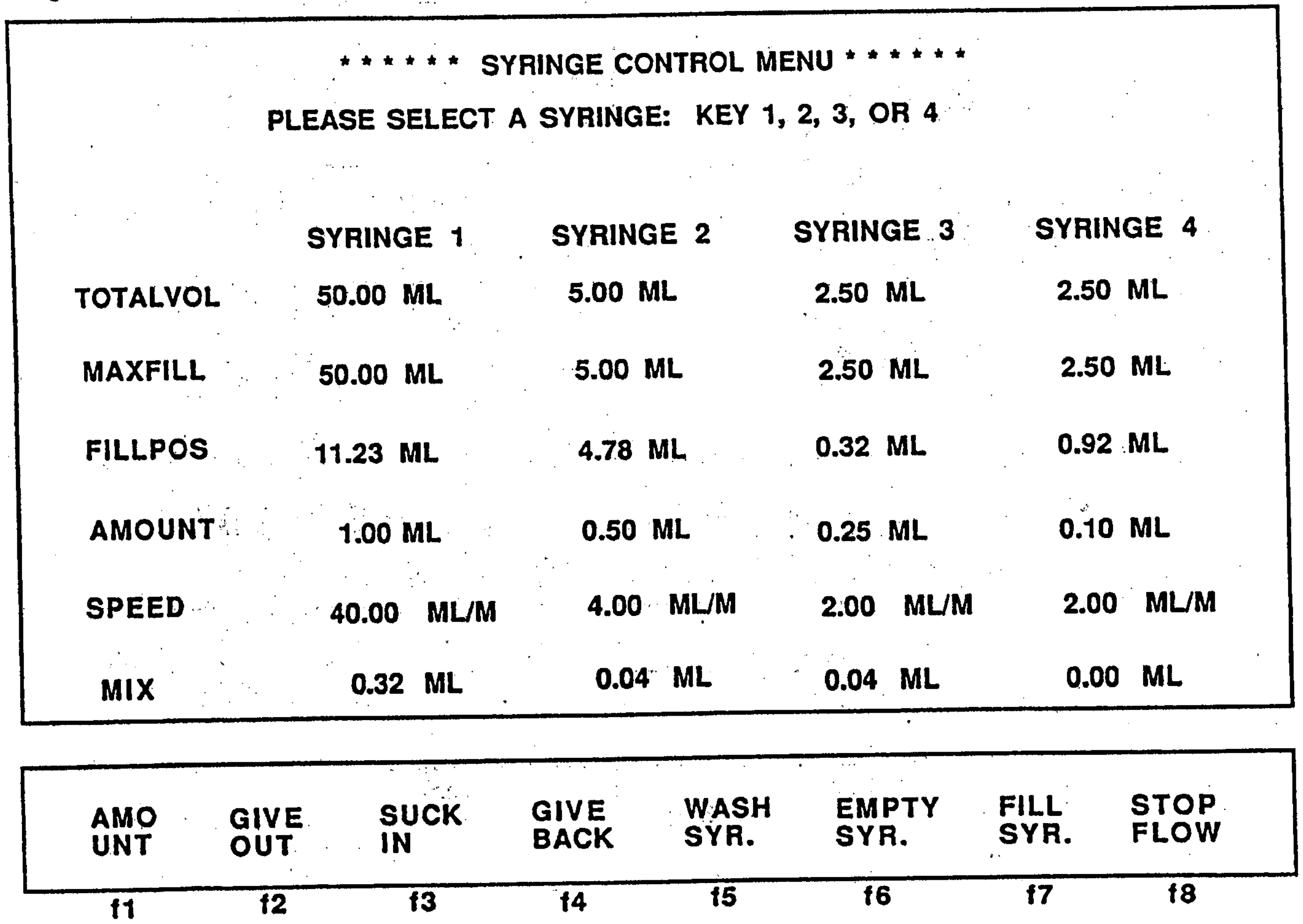

STATUS A
VALVES
221
STATUS B
SYRINGES
223
****** SYRINGE CONTROL MENU ******
S
SHIFT
PLEASE SELECT A SYRINGE: KEY 1, 2, 3, OR 4
1
ACTIVE SYRINGE
f1 f2 f3 f4 f5 f6 f7 f8
I A J B K C L D M E N F O G P H
- + * = Q 1 R 2 S 3 T 4
U 5 V 6 W 7 X 8
Y 9 Z 0
STOP
MENU SELECT KEYS
SHIFT

Fig. 8

Fig. 9



* * * * * * SYRINGE CONTROL MENU * * * * * *

PLEASE SELECT A SYRINGE: KEY 1, 2, 3, OR 4

	SYRINGE 1	SYRINGE 2	SYRINGE 3	SYRINGE 4
TOTALVOL	50.00 ML	5.00 ML	2.50 ML	2.50 ML
MAXFILL	50.00 ML	5.00 ML	2.50 ML	2.50 ML
FILLPOS	11.23 ML	4.78 ML	0.32 ML	0.92 ML
AMOUNT	1.00 ML	0.50 ML	0.25 ML	0.10 ML
SPEED	40.00 ML/M	4.00 ML/M	2.00 ML/M	2.00 ML/M
MIX	0.32 ML	0.04 ML	0.04 ML	0.00 ML

AMO UNT	GIVE OUT	SUCK IN	GIVE BACK	WASH SYR.	EMPTY SYR.	FILL SYR.	STOP FLOW
f1	f2	f3	f4	f5	f6	f7	f8

SUBSTITUTE SHEET

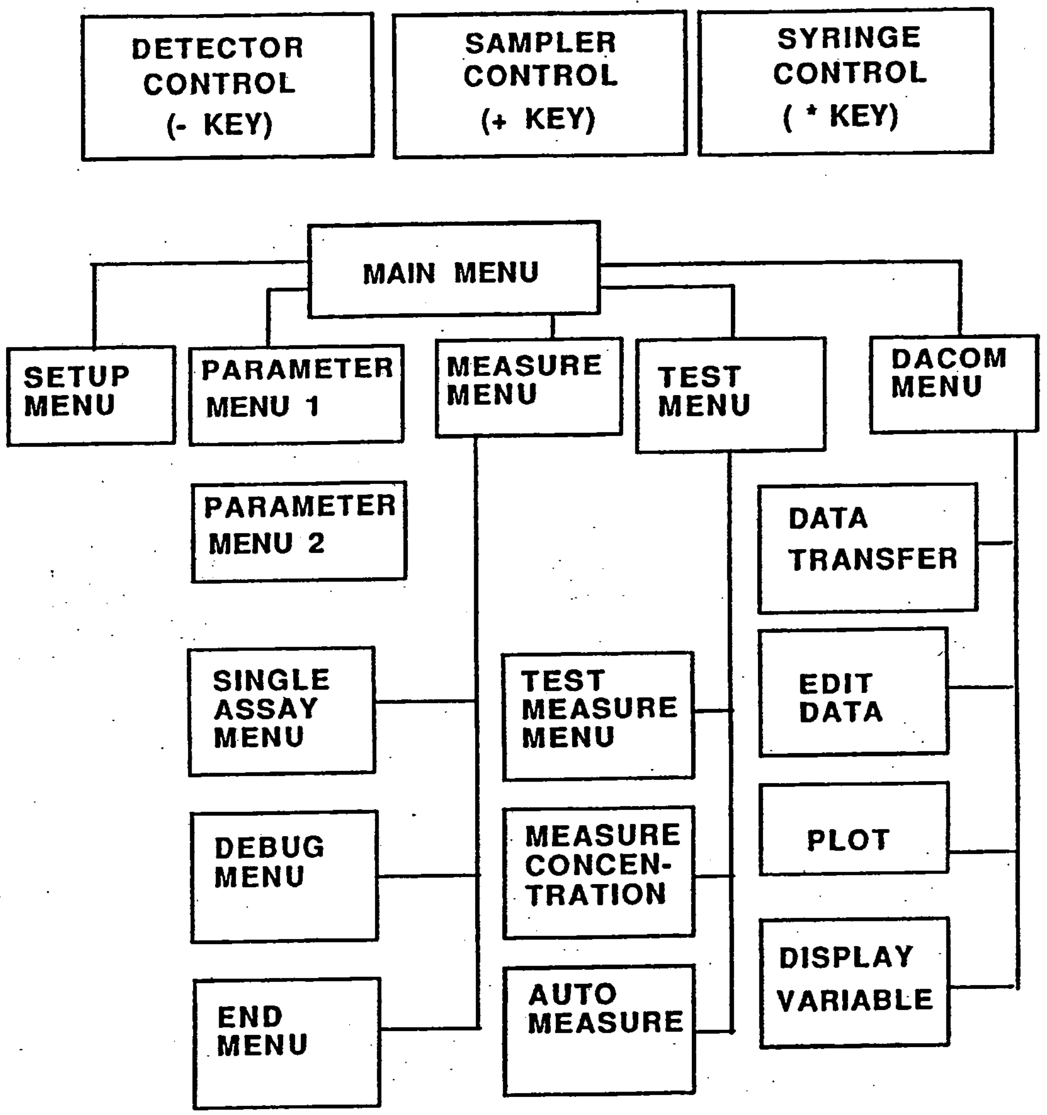
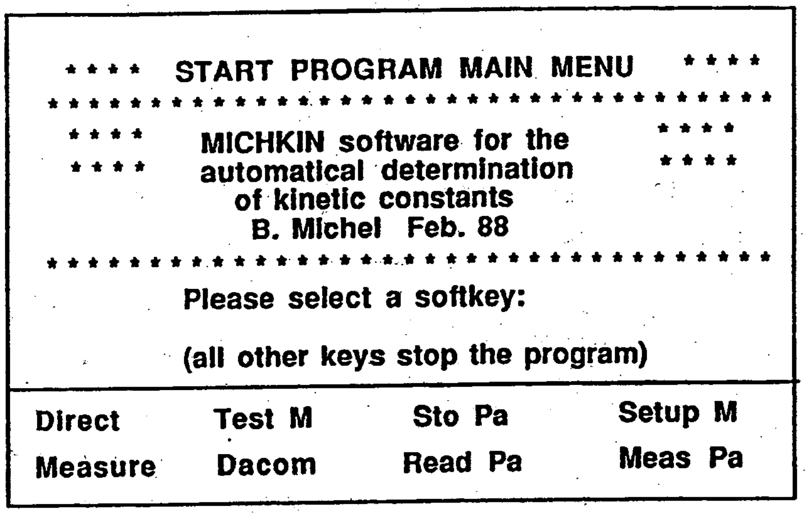

DETECTOR CONTROL (- KEY)
SAMPLER CONTROL (+ KEY)
SYRINGE CONTROL (* KEY)
MAIN MENU
SETUP MENU
PARAMETER MENU 1
PARAMETER MENU 2
MEASURE MENU
TEST MENU
DACOM MENU
SINGLE ASSAY MENU
DEBUG MENU
END MENU
TEST MEASURE MENU
MEASURE CONCEN-TRATION
AUTO MEASURE
DATA TRANSFER
EDIT DATA
PLOT
DISPLAY VARIABLE

Fig. 10 (A)



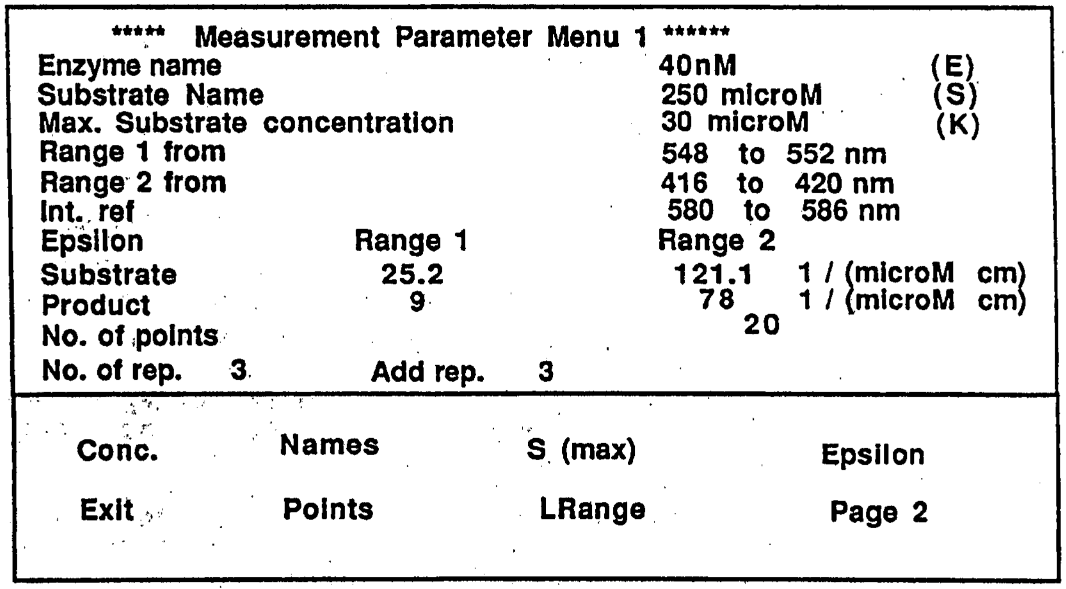
```
****  START PROGRAM MAIN MENU  ****
************************************
****   MICHKIN software for the   ****
****   automatical determination   ****
         of kinetic constants
         B. Michel  Feb. 88
************************************
      Please select a softkey:

   (all other keys stop the program)

Direct     Test M     Sto Pa     Setup M
Measure    Dacom      Read Pa    Meas Pa
```

Fig. 10 (B)

Fig. 10 (C)

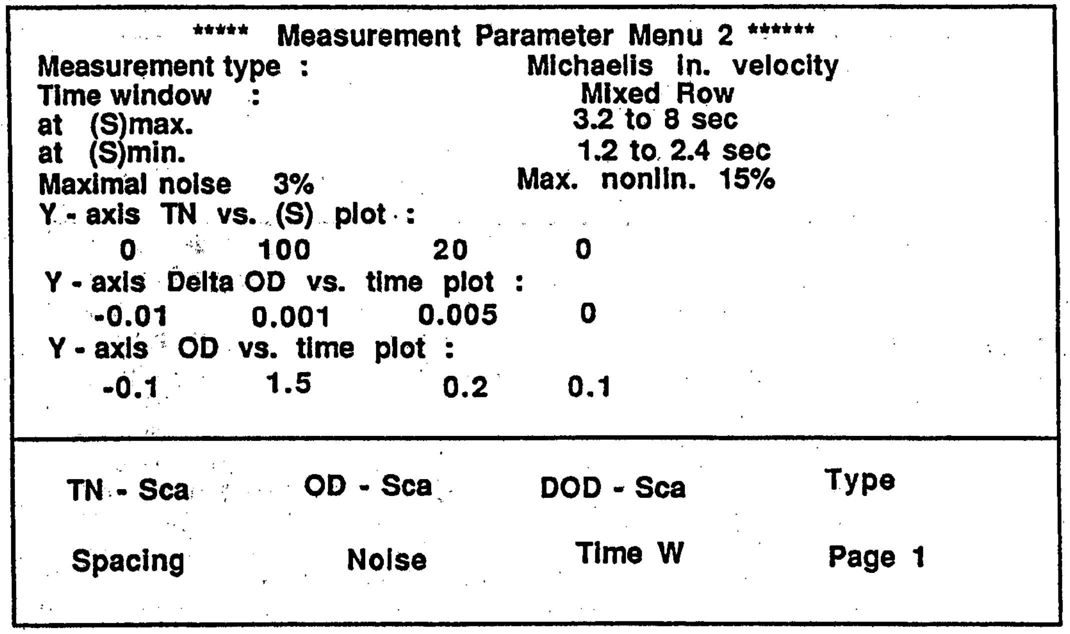


```
     *****  Measurement Parameter Menu 1 ******
Enzyme name                                40nM          (E)
Substrate Name                             250 microM    (S)
Max. Substrate concentration               30 microM     (K)
Range 1 from                               548  to  552 nm
Range 2 from                               416  to  420 nm
Int. ref                                   580  to  586 nm
Epsilon                Range 1             Range 2
Substrate               25.2                121.1   1 / (microM  cm)
Product                  9                   78     1 / (microM  cm)
No. of points                                  20
No. of rep.    3        Add rep.    3

Conc.          Names            S (max)          Epsilon

Exit           Points           LRange           Page 2
```

Fig. 10 (D)

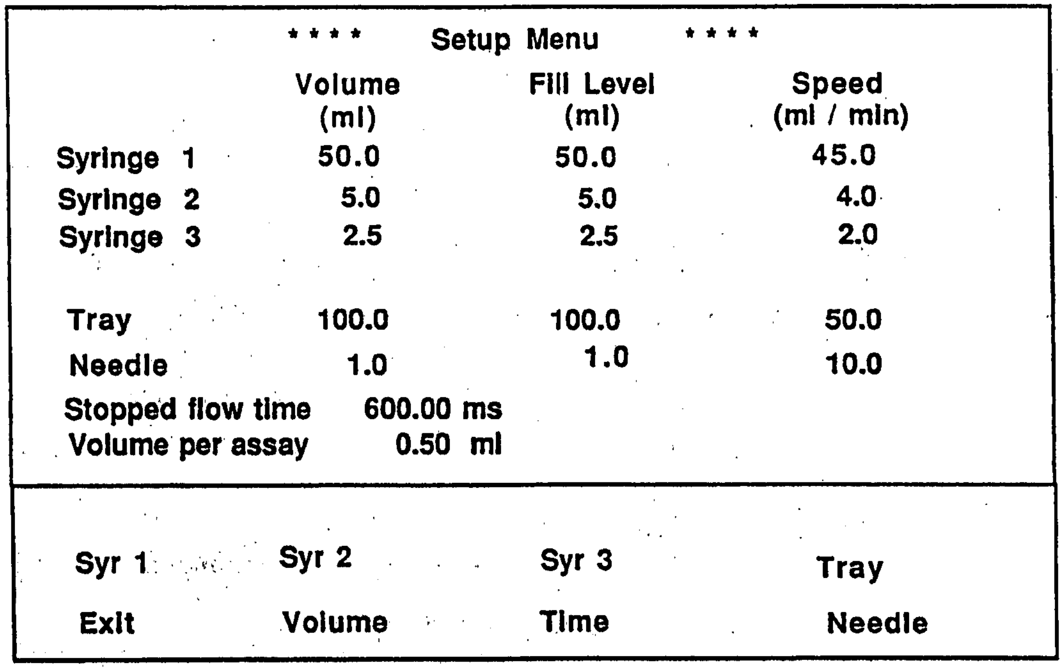


```
*****  Measurement Parameter Menu 2 ******
Measurement type  :                  Michaelis  in.  velocity
Time window     :                         Mixed  Row
at  (S)max.                                 3.2 to 8 sec
at  (S)min.                                  1.2 to 2.4 sec
Maximal noise    3%                  Max.  nonlin.  15%
Y - axis  TN  vs.  (S)  plot  :
     0          100          20         0
Y - axis  Delta OD  vs.  time  plot  :
    -0.01       0.001        0.005      0
Y - axis  OD  vs.  time  plot  :
    -0.1         1.5          0.2       0.1

 TN - Sca        OD - Sca        DOD - Sca        Type

 Spacing          Noise          Time  W          Page  1
```

Fig. 10 (E)

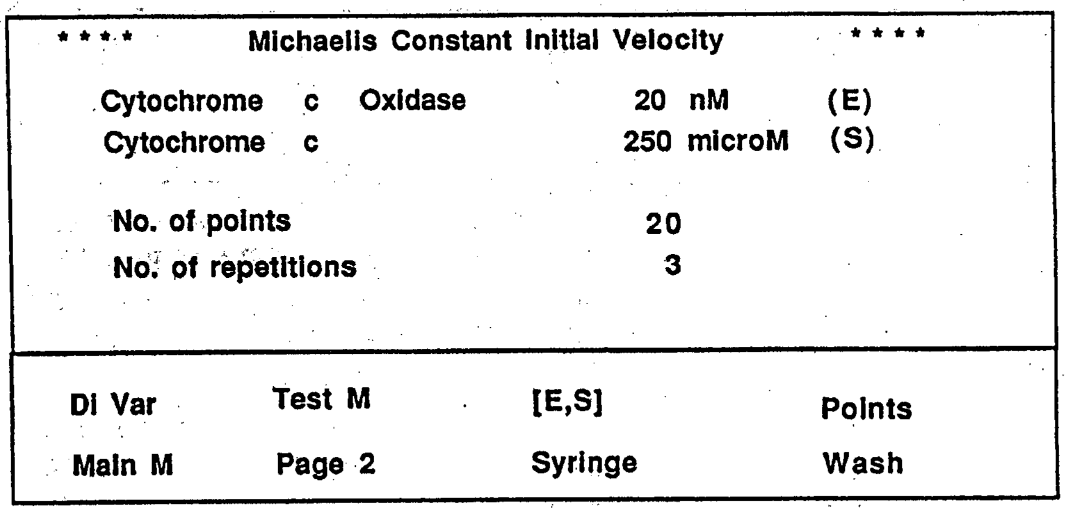


```
          ****    Setup Menu    ****
               Volume      Fill Level      Speed
                (ml)          (ml)       (ml / min)
Syringe 1       50.0          50.0          45.0
Syringe 2        5.0           5.0           4.0
Syringe 3        2.5           2.5           2.0

Tray           100.0         100.0          50.0
Needle           1.0           1.0          10.0
Stopped flow time   600.00 ms
Volume per assay      0.50 ml

Syr 1        Syr 2        Syr 3        Tray
Exit         Volume       Time         Needle
```

Fig. 10 (F)



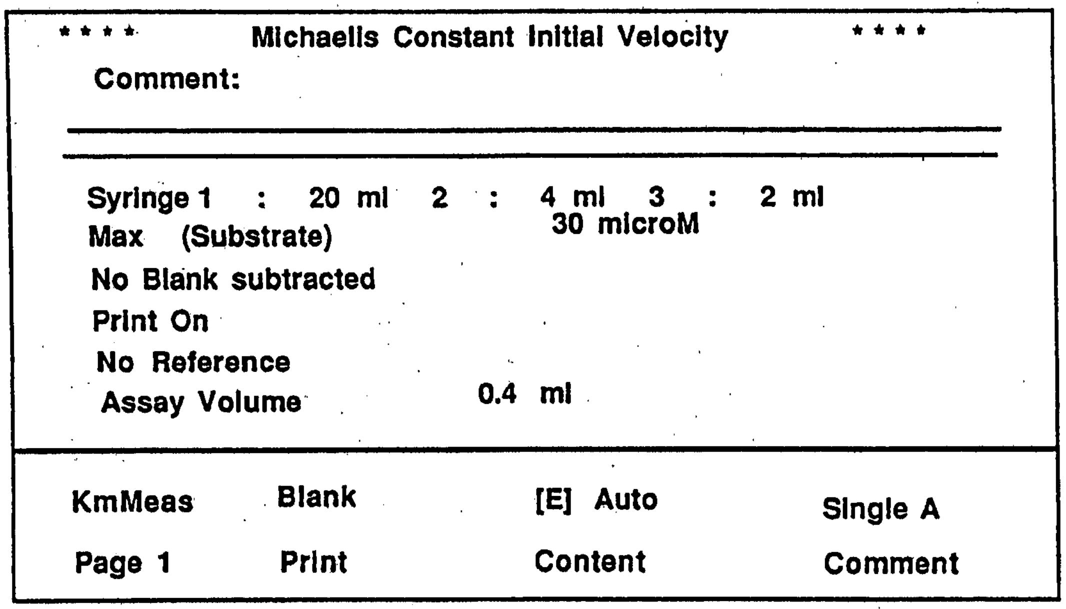
```
****         Michaelis Constant Initial Velocity        ****

   Cytochrome   c   Oxidase          20  nM      (E)
   Cytochrome   c                   250  microM  (S)

   No. of points                     20
   No. of repetitions                 3

Di Var       Test M        [E,S]           Points
Main M       Page 2        Syringe         Wash
```

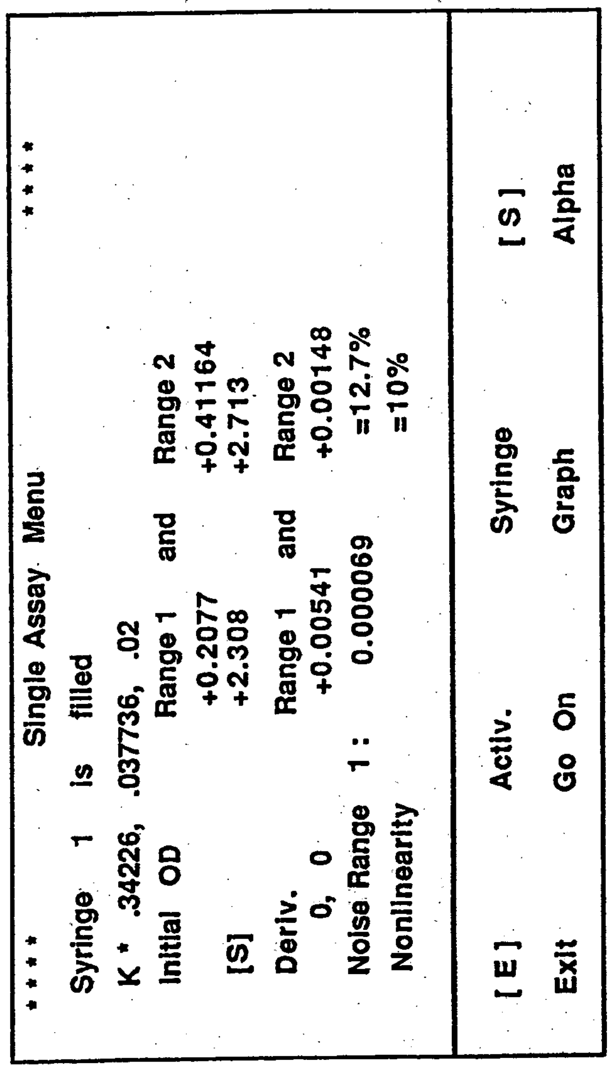
Fig. 10 (G)



```
****         Michaelis Constant Initial Velocity         ****
   Comment:
  ____________________________________________________________
  ____________________________________________________________

   Syringe 1   :   20 ml   2   :   4 ml   3   :   2 ml
   Max   (Substrate)                30 microM
   No Blank subtracted
   Print On
   No Reference
   Assay Volume                0.4  ml
---------------------------------------------------------------
  KmMeas       Blank          [E]  Auto          Single A

  Page 1       Print          Content            Comment
```

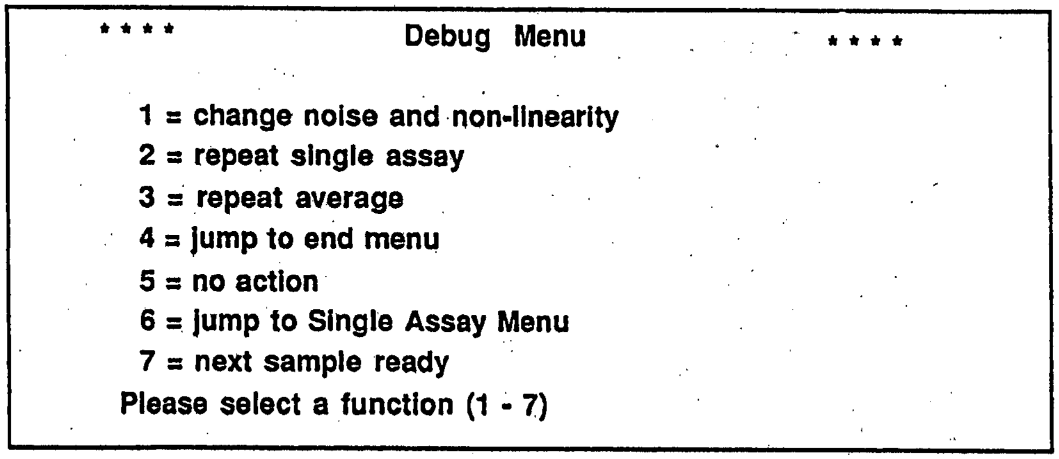


```
* * * *          Single Assay Menu              * * * *
Syringe  1  is  filled
K *  .34226, .037736,  .02
Initial OD        Range 1    and    Range 2
                  +0.2077           +0.41164
[S]               +2.308            +2.713
Deriv.            Range 1    and    Range 2
   0,  0          +0.00541          +0.00148
Noise Range  1 :    0.000069         =12.7%
Nonlinearity                         =10%

[ E ]        Activ.        Syringe        [ S ]
Exit         Go On         Graph          Alpha
```

Fig. 10 (H)

Fig. 10 (l)



```
****                Debug  Menu                ****

   1 = change noise and non-linearity
   2 = repeat single assay
   3 = repeat average
   4 = jump to end menu
   5 = no action
   6 = jump to Single Assay Menu
   7 = next sample ready
 Please select a function (1 - 7)
```

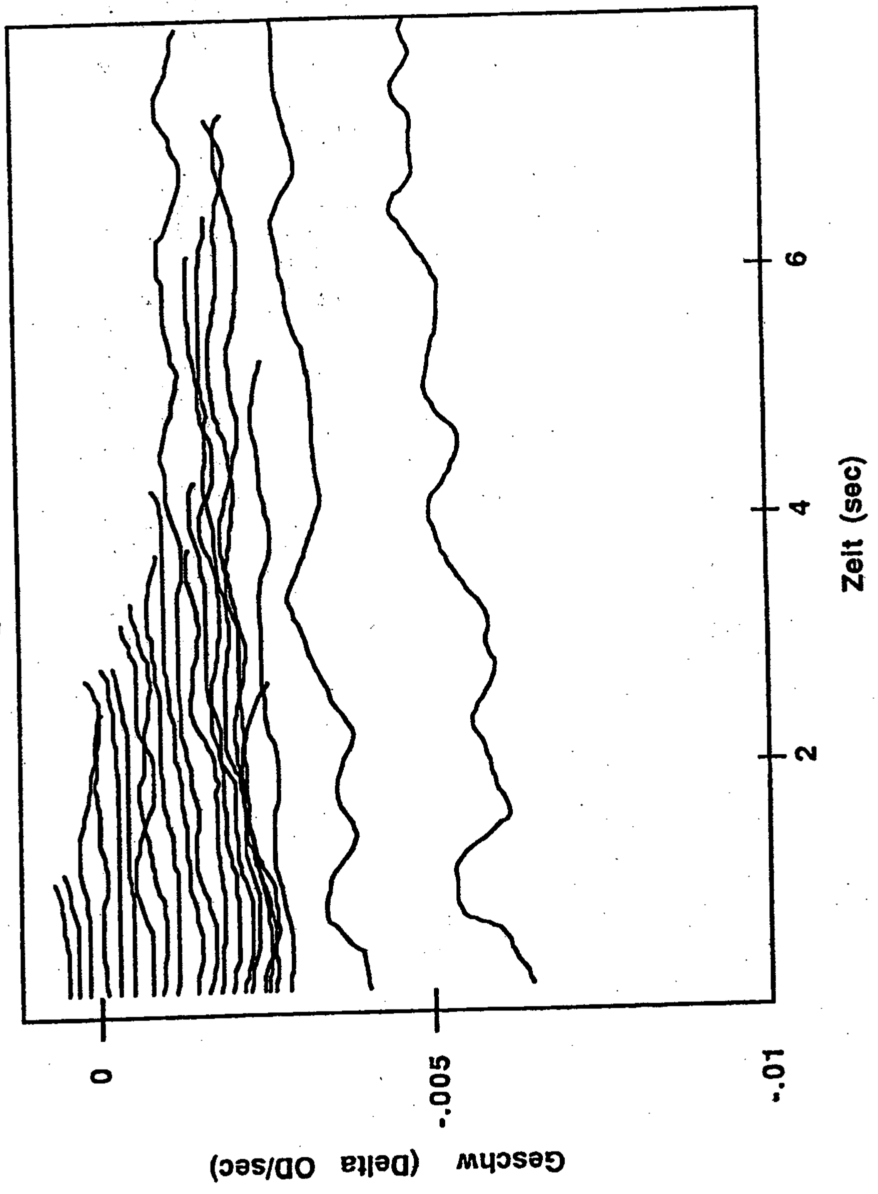
Measurement :	HORSCC B 5
6. Nov.	Volume : Modcyt
Michaelis constant determination	
Time window:	
at [S]min	.4 to 1.6 sec
at [S]max	3.2 to 8 sec
Interval	.4 integration .4
Maximal Noise	5%
λ -range 1 from	549 to 552 nm
λ -range 2 from	418 to 420 nm
Internal ref. from	580 to 586 nm
[cytochrome oxidase]	:. 30 nm
conc. in assay	: 3 nm
[cytochrome c]	: 250 μm
Max. conc. in assay	: 30 μm
No. of points 20	Rep. 3
Delta epsilon 1	: -16.2 1/(mM cm)
Delta epsilon 2	: -43.1 1/(mM cm)
Assay volume	0.4 ml

Fig. 11

No.	Range 1	Range 2	Noise
0	-.000029	-.000114	-.000014
1	-.000153	-.000178	-.000024
2	-.000399	-.001021	-.000012
3	-.000559	-.001460	-.000023
4	-.000655	-.001778	-.000028
5	-.000797	-.002117	-.000031
6	-.000914	-.002565	-.000029
7	-.001067	-.002885	-.000036
8	-.001206	-.003216	-.000051
9	-.001281	-.003468	-.000013
10	-.001402	-.003778	-.000031
11	-.001550	-.004213	-.000045
12	-.001706	-.004633	-.000059
13	-.001937	-.005262	-.000053
14	-.002094	-.005669	-.000037
15	-.002317	-.006200	-.000058
16	-.002529	-.006613	-.000042
17	-.002700	-.006712	-.000055
18	-.002817	-.005955	-.000054
19	-.003169	-.004821	-.000146
20	-.003628	-.003112	-.000150

Fig. 12

FIG. 13

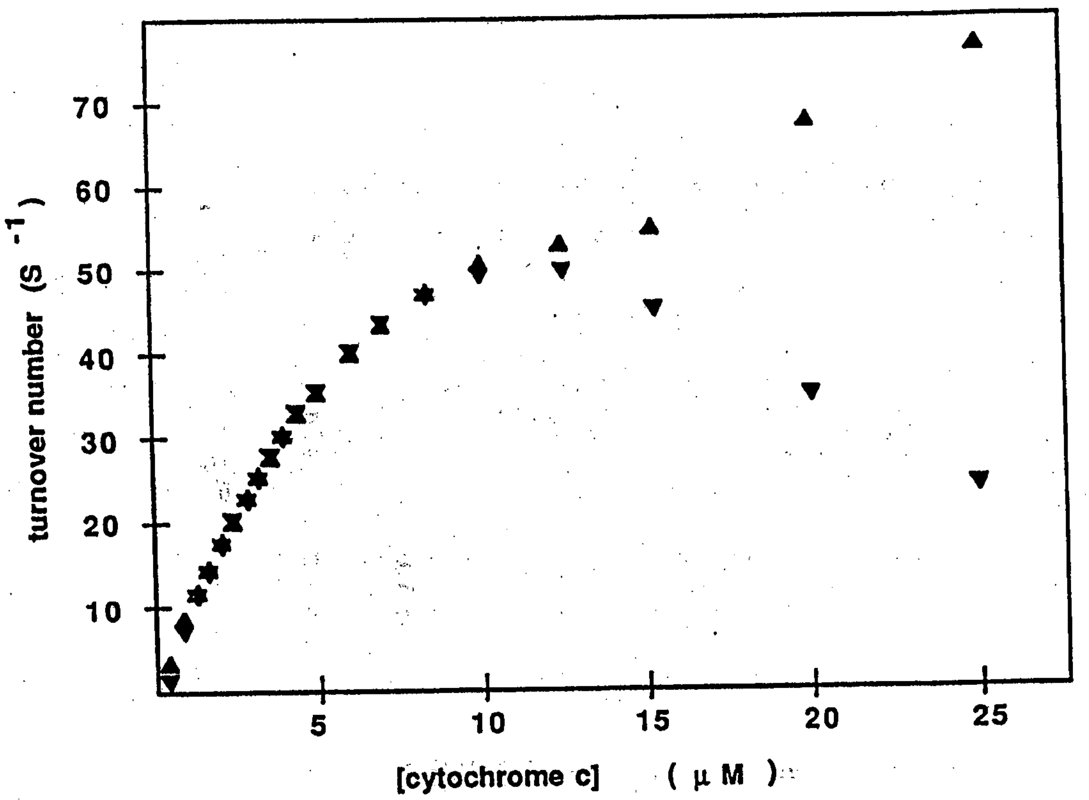


Measurement : HORSCC B 5
6. Nov. Volume : Modcyt

Michaelis constant of Oxidase VIA
in 0.2% Tween 80 at 150 mM I.str.

No.	Range 1 TN(l/s)	Range 2 TN(l/s)	[Substr] µM
0	+.6	+.9	0.00
1	+3.2	+1.4	0.47
2	+8.2	+7.9	0.69
3	+11.5	+11.3	0.92
4	+13.5	+13.7	1.16
5	+16.4	+16.4	1.42
6	+18.8	+19.8	1.71
7	+22.0	+22.3	2.03
8	+24.8	+24.9	2.40
9	+26.4	+26.8	2.82
10	+28.8	+29.2	3.32
11	+31.9	+32.6	3.93
12	+35.1	+35.8	4.67
13	+39.8	+40.7	5.59
14	+43.1	+43.8	6.74
15	+47.7	+48.0	8.21
16	+52.0	+51.1	10.08
17	+55.6	+51.9	12.49
18	+58.0	+46.1	15.62
19	+65.2	+37.3	19.69
20	+74.6	+24.1	25.00

Fig. 14



Maximal turnover	85.65 l/s
Michaelis constant	6.232 μ M
Correlation	0.9981

Sample Michaelis constant determination

Fig. 15

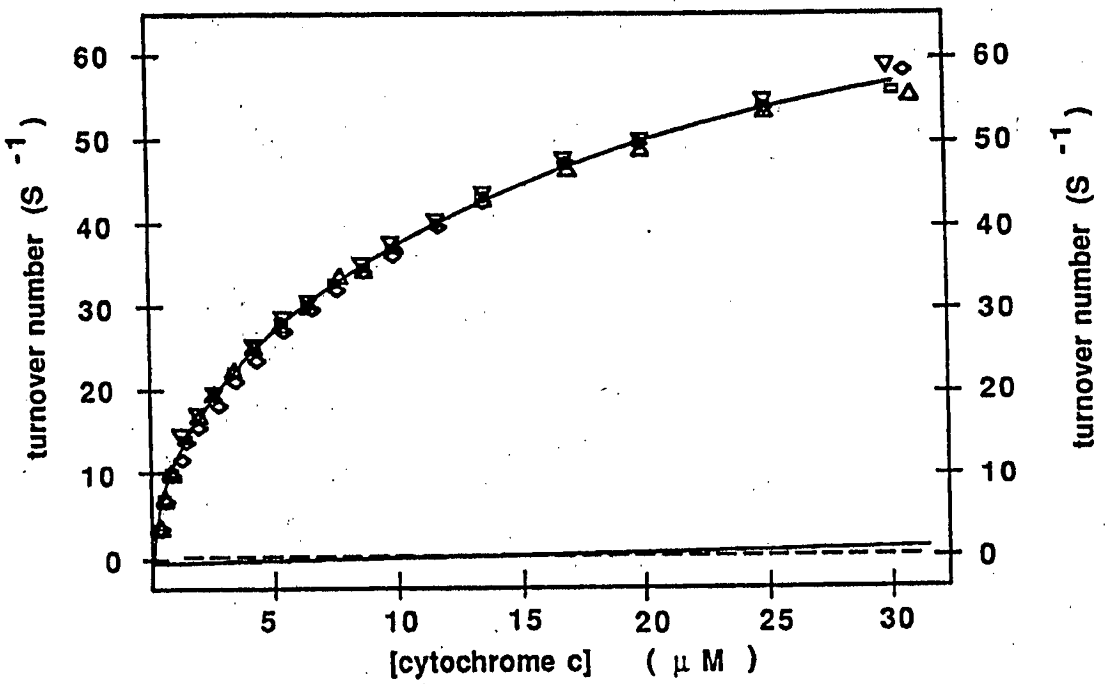


Fig. 16 Reproducibility and noise of assay series (Δ,∇,$\Diamond$,⊟) four series of assays measured under identical conditions. (———) average of these four series. Blank reaction of reduced cytochrome c without oxidase but otherwise same conditions (········· , three dimensions).

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US89/04981

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC(5): G01N 21/05; G01N 35/00
US CL : 422/67,81; 436/34,52

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
U.S.	422/62,67,68,81; 436/34,52,55 364/499,500

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X Y	US, A, 3,960,707 (GROSS), PUBLISHED 01 JUNE 1976. SEE ENTIRE DOCUMENT	9,10 1-8,11
A	US, A, 3,932,136 (STICKNEY), PUBLISHED 13 JANUARY 1976. SEE ENTIRE DOCUMENT	4
A	US, A, 4,263,406 (BOSTICK), PUBLISHED 21 APRIL 1981. SEE ENTIRE DOCUMENT	1-11
A	US, A, 4,326,940 (ECKLES), PUBLISHED 27 APRIL 1982. SEE ENTIRE DOCUMENT	1-11
A	US, A, 4,399,101 (QUEEN), PUBLISHED 16 AUGUST 1983. SEE ENTIRE DOCUMENT	4
A	US, A, 4,612,289 (FURUTA), PUBLISHED 16 SEPTEMBER 1986	1-11

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search 08 FEBRUARY 1990	Date of Mailing of this International Search Report 06 MAR 1990
International Searching Authority ISA/US	Signature of Authorized Officer [illegible] MICHAEL S. MARCUS